



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*

# RISC-V principles for understanding how to freely develop new solutions

November 2024

BETC RISC-V Principles



# Who we are?

## Day 1



### **Teresa Cervero**

Group leader

Technical Management HW engineering

- *Course Convener*



### **Xavier Martorell**

Group leader

Parallel Programming Models Group

- *He will lecture Fundamentals and OS session*



### **Rohan Ahmed**

Research Engineer

Parallel Programming Models Group

- *He will assist you on the Fundamentals and OS session*

## Day 2

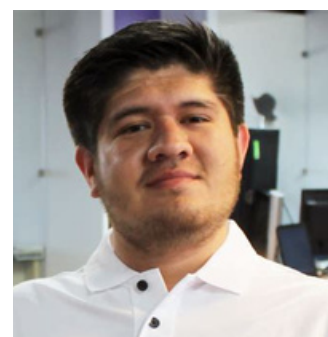


### **Aaron Call**

Established researcher

Data Centric Computing Group

- *He will lecture Virtualization for cloud session*



### **Julián Pavón**

Research Engineer

Computer Architecture for Parallel Paradigms Group

- *He will lecture the Potential of custom instructions session*



### **Iván Vargas**

Research Engineer

Computer Architecture for Parallel Paradigms Group

- *He will lecture the Potential of custom instructions session*

## Day 3



### **Roger Ferrer**

Group leader

Compiler and Tool chain for HPC Group

- *He will lecture RVV with the compiler session*



### **Filippo Mantovani**

Group leader

Software Development Vehicles

- *He will lecture Parallelization and RVV session*



### **Pablo Vizcaíno**

Research Engineer

Software Development Vehicles Group

- *He will assist you on the RVV hands-on session*

# Training structure

Time
9:00-10:00
10:00-11:00
11:00-12:00
12:00-13:00
13:00-14:00
14:00-15:00
15:00-16:00
16:00-17:30

Day 1: Fundamentals
<i>RISC-V ecosystem</i>
<i>RISC-V ISA Basics</i>
Lunch Break
<i>Booting an OS with QEMU</i>
Break (15:30 - 16:00)
<i>Hands-on</i>

Day 2: virtualization & emulation
<i>RISC-V &amp; Pytorch</i>
<i>RISC-V &amp; Singularity</i>
Break (11:00 - 11:30)
<i>Hands-on</i>
Lunch Break
<i>RISC-V custom instructions</i>
Break (15:30 - 16:00)
<i>Hands-on</i>

Day 3: RISC-V vector Extension
<i>HPC Fundamentals</i>
<i>RISC-V Vector extension</i>
Break (11:00 - 11:30)
<i>Hands-on</i>
Lunch Break
<i>RVV &amp; compiler</i>
Break (15:30 - 16:00)
<i>Hands-on</i>

# Software requirements

## SSH: Secure shell (HPC system connection)

- **Linux/Mac OS:** native support of secure shell “ssh user@host”
- **Windows:** need to install a ssh program
  - PuTTY: <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
  - MobaXterm: <http://mobaxterm.mobatek.net/download.html>

QEMU: [www.qemu.org](http://www.qemu.org)

RISCV Documentation: <https://wiki.qemu.org/Documentation/Platforms/RISCV>



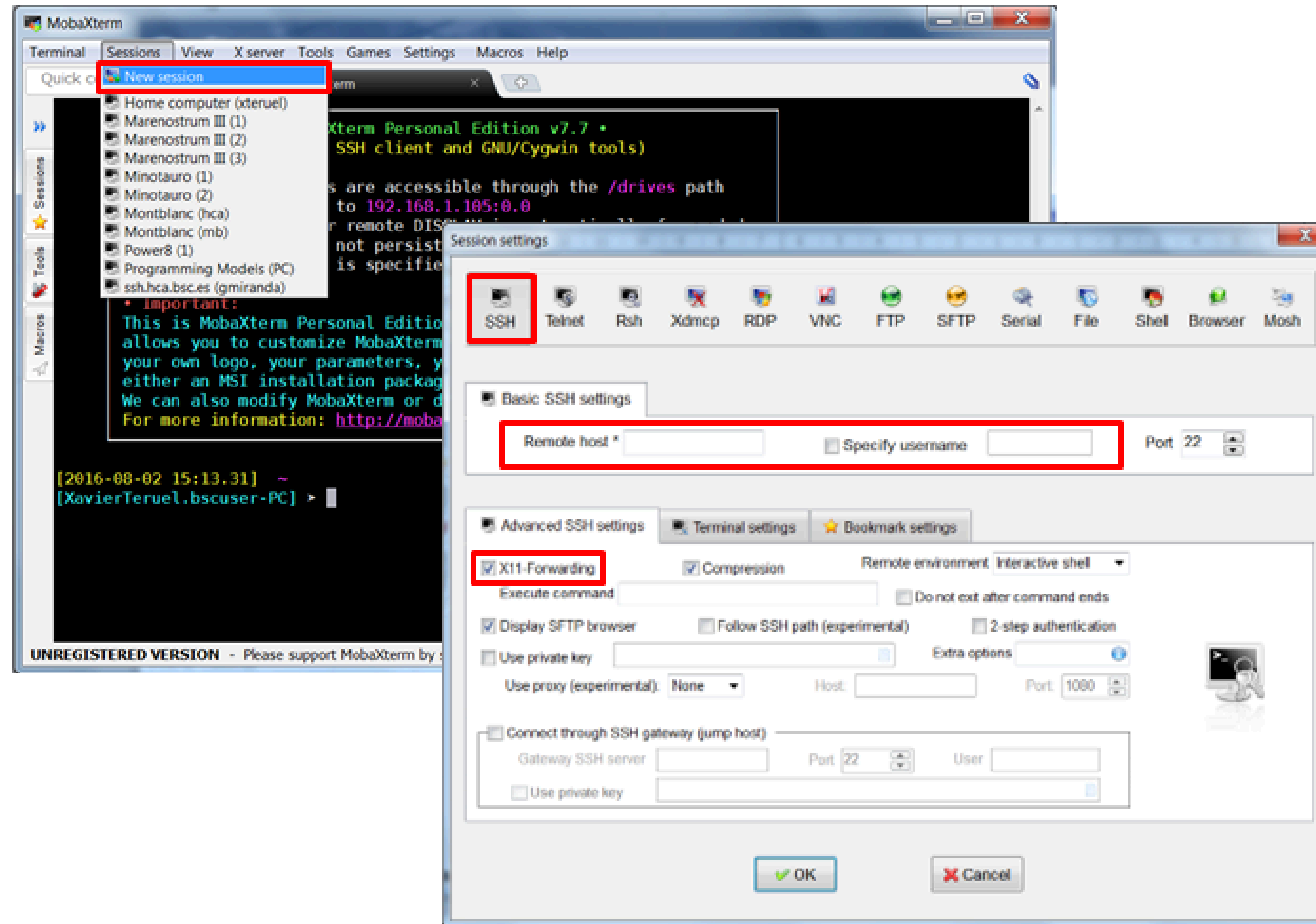
**QEMU**  
A generic and open source machine emulator and virtualizer

Full-system emulation  
Run operating systems for any machine, on any supported architecture

User-mode emulation  
Run programs for another Linux/BSD target, on any supported architecture

Virtualization  
Run KVM and Xen virtual machines with near native performance

# MobaXterm: ssh configuration



# Training material

- Slides available after sessions at:  
<https://tinyurl.com/betc-riscv-2024>
- Hands-on: connection to HCA network (BSC)  
Temporal accounts  
Guidelines per session
- Communication by email  
Topic: *[BETC-RISCV]* <Query/Doubt>  
To: [teresa.cervero@bsc.es](mailto:teresa.cervero@bsc.es)



# RISC-V: An overview



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*

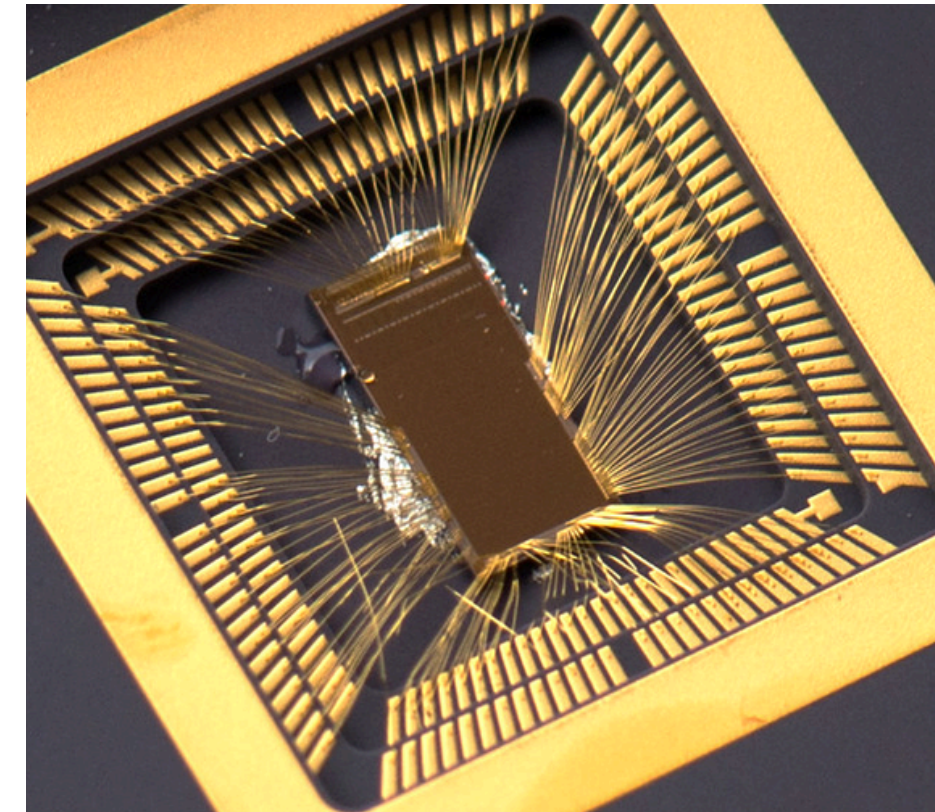
# Global standards: a catalyst to accelerate technical Innovation



Standards have been critical to technology innovation, adoption, and growth for decades



Standards create access to opportunities and spur growth for a wide range of stakeholders

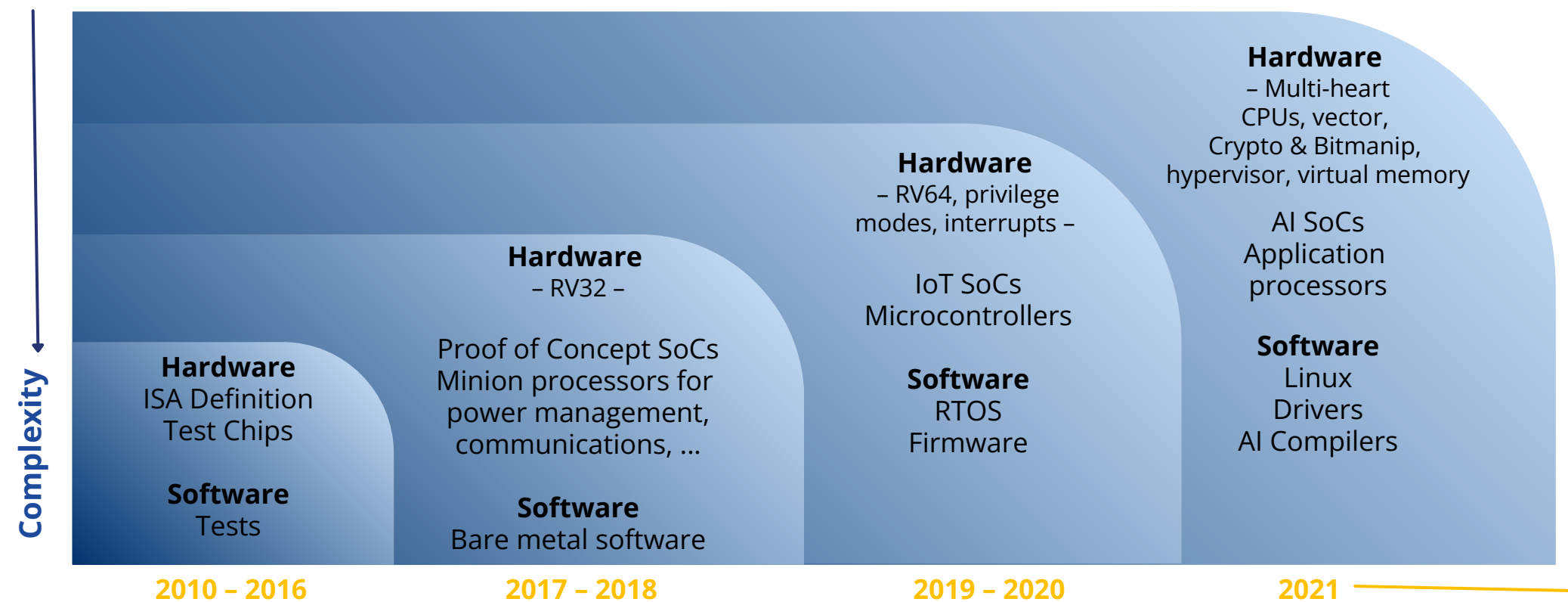


**RISC-V** is a **standard-defined Instruction Set Architecture** developed by a global community



# What is RISC-V?

- An open **I**nstruction **S**et **A**rchitecture (ISA)
  - Standardisation activities driven by expert members
  - Numerous areas of focus ranging from HPC & ML to the data centre to embedded computing
    - Additional extensions, tools and resources



- Why is **RISC-V** interesting **for HPC**?
  - Freely develop new technologies based upon RISC-V
  - Standards community driven
  - Modular design of ISA
    - Pick and mix parts that you need
    - from low power embedded devices to HPC and IA accelerators

# Why RISC-V International?

## RISC-V International (RVI): a global non-profit association

- Founded in 2015
- Community driven organization:  
[4000+ members](#) in + 70 countries  
Across industries and technical disciplines

### Functions:

- Supports the open RISC-V Instruction Set Architecture (ISA)
  - Develops additional extensions, tools, and resources paving the way for the next 50 years of computing design and innovation
  - Connects the community and industry through academia, commercialization, and strategic leadership
- RISC-V is not...**
- A company
  - A CPU implementation



# The definition of open computing is RISC-V

RISC-V is the most prolific and open Instruction Set Architecture in history

## RISC-V is inevitable

Mission: RISC-V is the industry standard ISA across computing

- **>10 Billion RISC-V cores** already shipped
- Adoption moving rapidly **across all domains**
- Demand at every performance level
- Shared investment driving ecosystem

## RISC-V enables the best processors

RISC-V enables profound innovation from low end to high end applications

- Inherent and sustainable performance and efficiency advantage
- Design **flexibility and freedom**
- Supported by massive community enabling the most efficient designs for **full spectrum of applications**
- Modern design for **fewer instructions**

## RISC-V is rapidly building the strongest ecosystem

RISC-V instrumented with software top of mind

- **Open standards enable software choice** Applications keen to run on RISC-V.
- Toolchain and OS support required for Extension ratification
- **Single hypervisor standard** to simplify and unify application support
- Thousands of software developers
- Strategic **investment by industry and geographies**

# Challenges and opportunities

	Legacy ISA	RISC-V ISA
<b>Complexity</b>	1500+ base instructions	47 base instructions
<b>Design freedom</b>	\$\$\$ – Limited	Free – Unlimited
<b>License and Royalty fees</b>	\$\$\$	Free
<b>Design ecosystem</b>	Moderate	Growing rapidly. Numerous extensions, open and proprietary cores
<b>Software ecosystem</b>	Extensive	Growing rapidly

## Barriers removed

- ... Design risk
- ... Cost of entry
- ... Partner limitations
- ... Supply chain

# Current ISA business models

Business Model	Chips?	Architecture License	Commercial Core IP	Add Own Instructions	Open-Source Core IP
Microprocessor	Yes, <i>two</i> vendors	<b>No</b>	Yes, <i>one</i> vendor	<b>No</b>	<b>No</b>
Proprietary ISA	Yes, <i>many</i> vendors	Yes, <i>expensive and restricted</i>	Yes, <i>one</i> vendor	<b>No, (Mostly)</b>	<b>No</b>
RISC-V Open standard ISA	Yes, <i>many</i> vendors	Yes, ISA is an <i>open standard</i>	Yes, <i>many</i> vendors	<b>Yes</b>	<i>Yes, many available</i>

**RISC-V enables design freedom**

# Open interfaces are accepted practice

Field	Proprietary predecessor	Open Standard	Open Implementation	Commercial implementation on open standard
<b>Networking</b>	Now obsolete	Ethernet, TCP/IP	Many	Cisco, Juniper
<b>OS</b>	Windows	Posix	Linux, FreeBSD	Red Hat, Canonical, Suse, AIX, Zephyr
<b>Compilers</b>	Intel icc, ARMcc, Xcode	C	gcc, LLVM	Greenhills, IAR
<b>Databases</b>	Oracle 12C, DB2	SQL	MySQL, PostgreSQL	Oracle, SQLServer, DB2
<b>Graphics</b>	DirectX	OpenGL	Mesa3D	NVIDIA, AMD, Intel
<b>ISA</b>	x86, ARM, IBM360	<b>RISC-V</b>	LowRISC, other community led	Numerous RISC-V implementations

Successful open standards, enabling multiple implementations

# Industry outlook



## Cloud and

**data center applications** top cloud providers like Amazon and Alibaba are designing their own chips.



**Mobile and wireless** continue rapid evolution with each generation of hardware and increased capability.



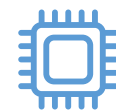
**Automotive** is transforming from autonomous vehicles to infotainment to safety, the whole vehicle relies on innovative electronics.



**Consumer and IoT devices** bring incredible innovation and volume with billions of connected devices being in the next 5-10 years.



**Industrial IoT** incorporating artificial intelligence in manufacturing and industrial processes.



**Memory** was the largest semiconductor category by sales with \$158 billion in 2018, and the fastest-growing, with sales increasing.

# RISC-V future is very bright

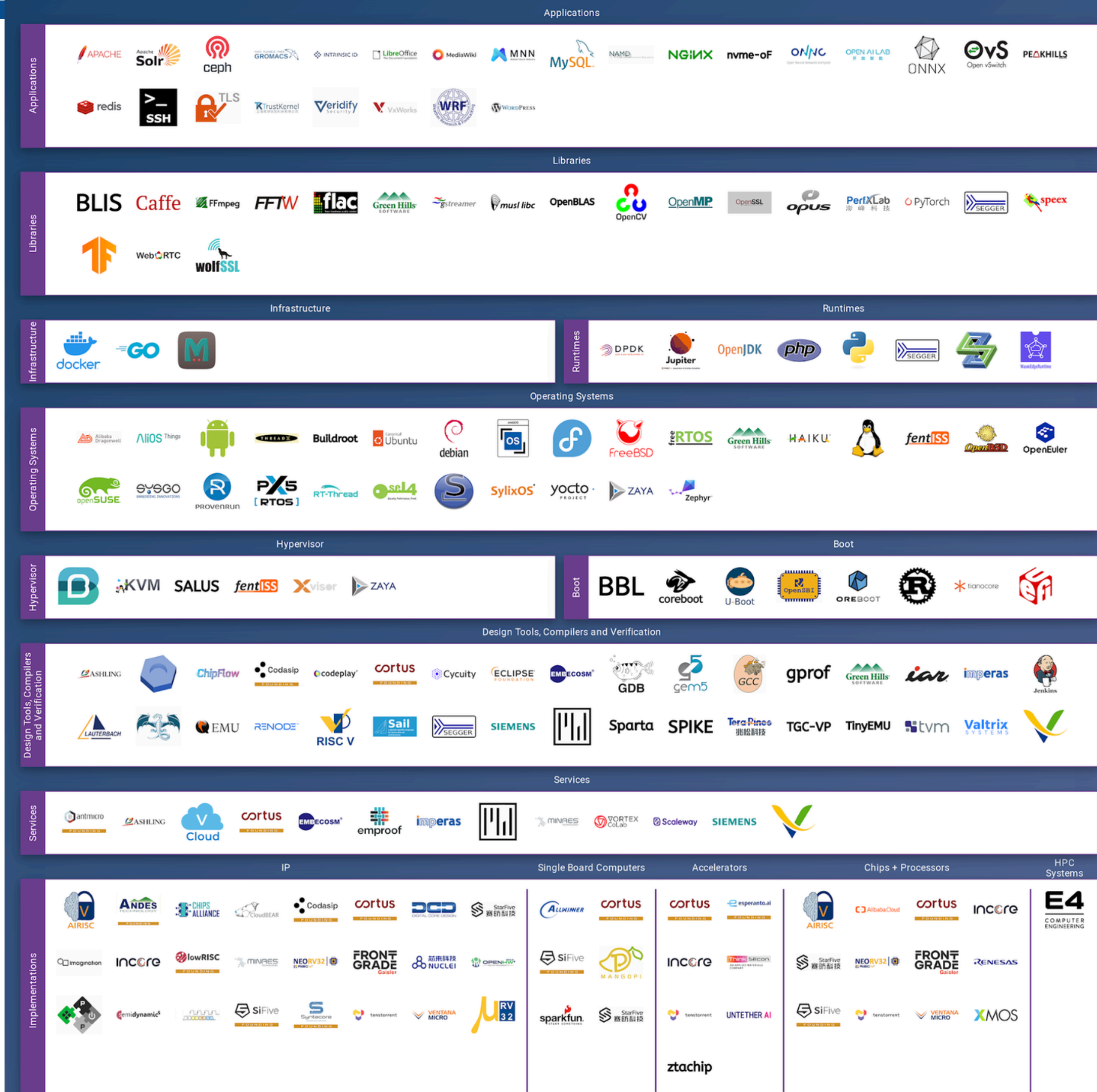
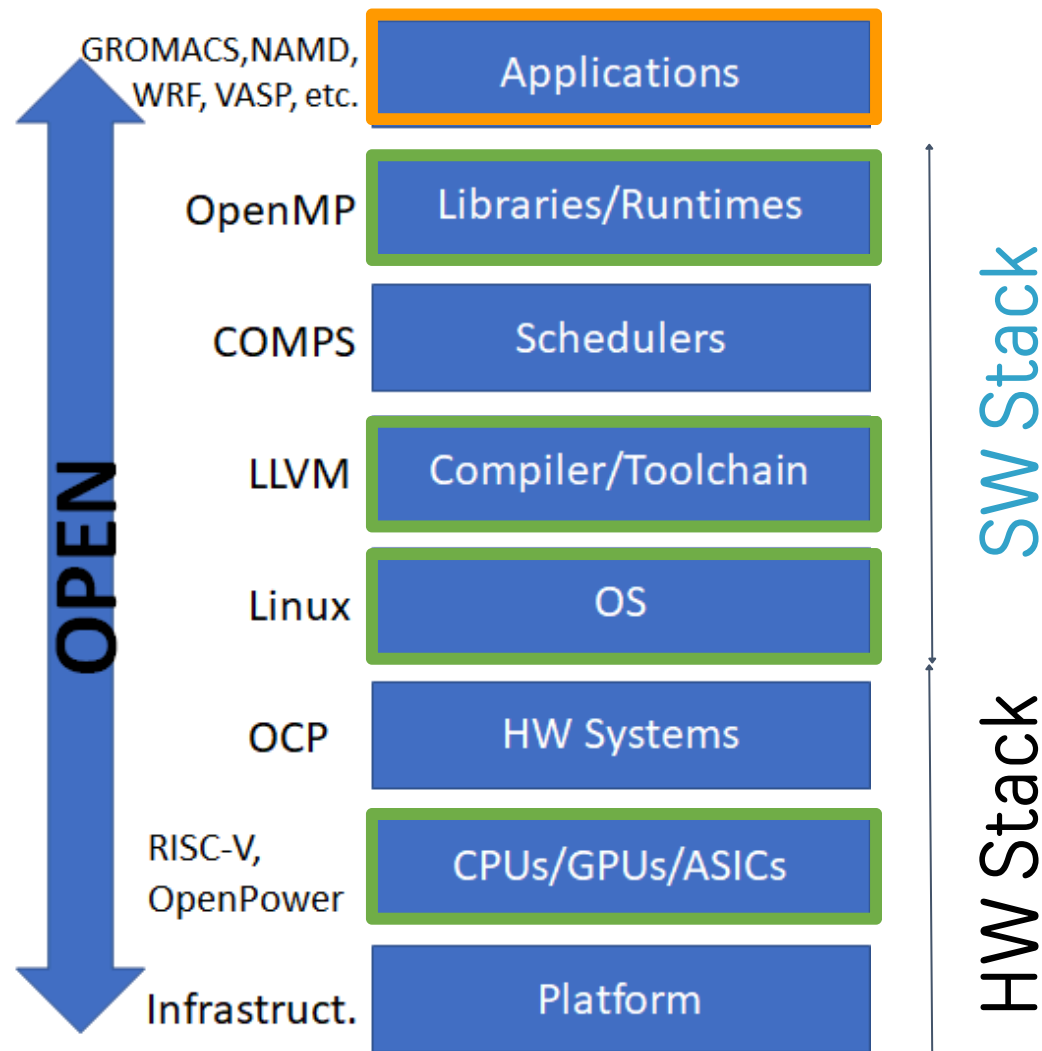


**Barcelona Supercomputing Center**  
Centro Nacional de Supercomputación



# RISC-V Ecosystem

- RISC-V Ecosystem landscape <https://landscape.riscv.org/>
  - Changing all the time
- RISC-V >< BSC ecosystem



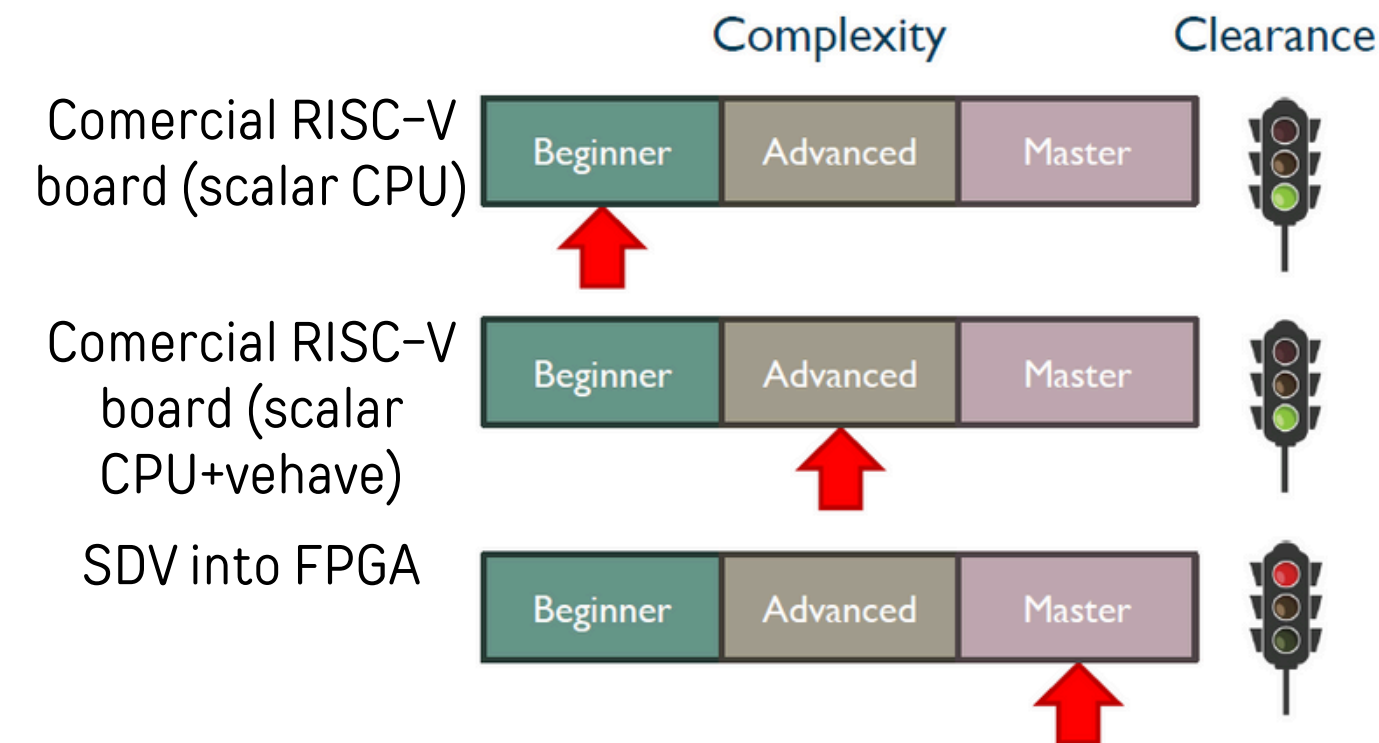


# BSC & RISC-V ecosystem



## • BSC contributions

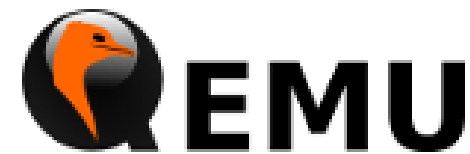
- Community Member of RVI: <https://riscv.org/members/>
- Participant in SIG & TW: <https://lists.riscv.org/g/main>
  - Special Interest Group (SIG): Forums to discuss
    - SIG-HPC → High-Performance Computing
    - SIG-Safety → Functional Safety
  - Technical Working Group (TW)
    - RVV (Vector Extension)
    - IME (Integrated Matrix Extension)
    - AME (Attached Matrix Extension)
- Contributing to strengthen the ecosystem
  - RISC-V Summit Europe: <https://riscv-europe.org/>
  - SOHA (Spanish Open Hardware Alliance): <https://sohariscv.org/>
  - Workshops and events (SC, ISC, HiPEAC...)
  - BSC LOCA GitHub: <https://github.com/bsc-loca>
  - FPGA-cluster: <https://www.bsc.es/supportkc/docs/MEEP/intro>



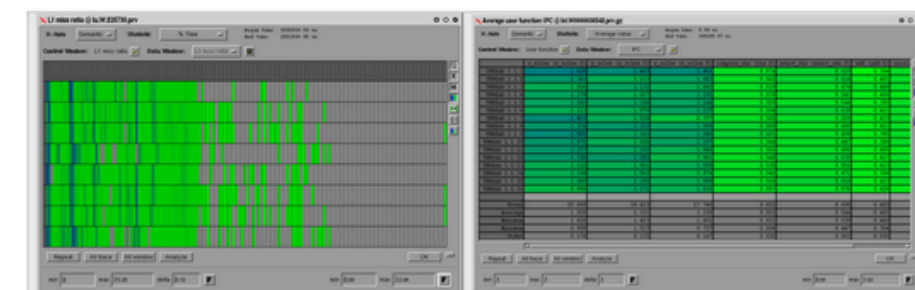
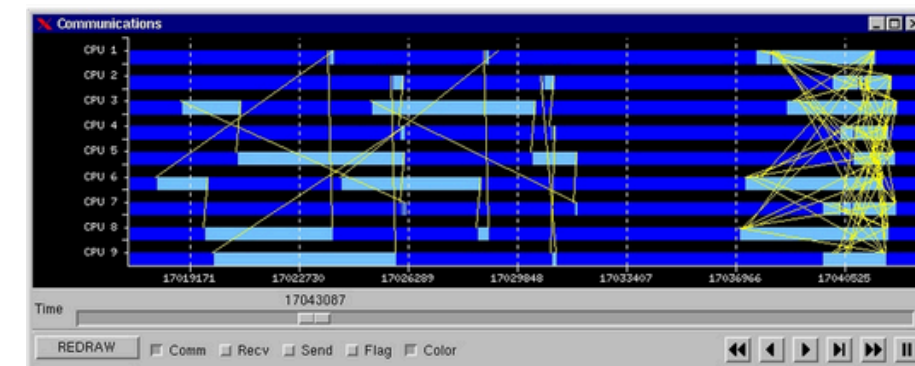
# BSC & RISC-V ecosystem

## • SW Stack Tools

- QEMU (BSC.RAVE): <https://www.qemu.org>
  - Gem5 (BSC.RISC-V): <https://github.com/gem5/gem5>
  - Pytorch (BSC.porting RV): <https://pytorch.org/>
  - PyCOMPS/COMPs: <https://pypi.org/project/pycompss/>
  - TensorFlow (BSC.porting RV): <https://www.tensorflow.org/>
  - OpenStack (BSC.porting RV): <https://www.openstack.org/>
  - OpenMP (BSC.target): <https://www.openmp.org/>
  - MPI (BSC.Vectorization): <https://docs.open-mpi.org/>
  - OpenSBI (images): <https://github.com/riscv-software-src/opensbi>
  - LLVM (BSC.autovectorization): <https://llvm.org/>
  - Vehave: <https://www.bsc.es/research-and-development/software-and-apps/software-list/vehave/downloads>
  - Paraver: <https://tools.bsc.es/paraver>
  - Extrae: <https://tools.bsc.es/extrae>
- More under construction...



openstack



# Sargantana core & environment

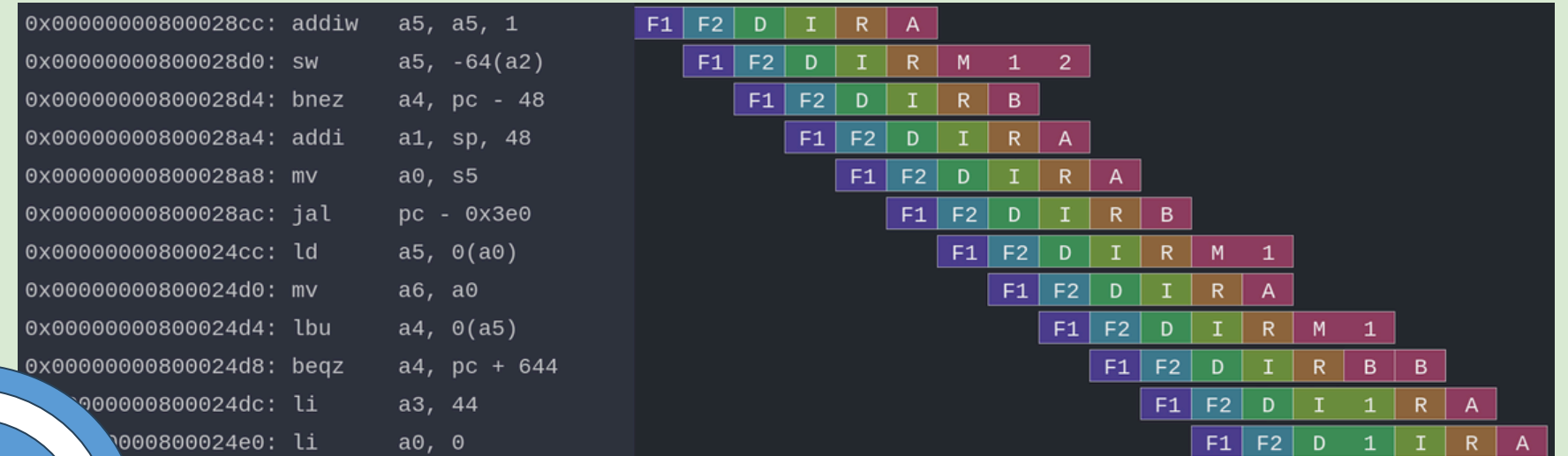
## Simulators support



Questa advanced simulator



## Pipeline visualizer (Konata support)



## Verification environment

### Gitlab CI/CD

- Verilator compilation
- ISA tests verification
- Microbenchmark performance verification

### Commit trace

- Spike compatible

### UVM environment (WIP)

- Spike co-simulation
- RISC-V DV random test generator



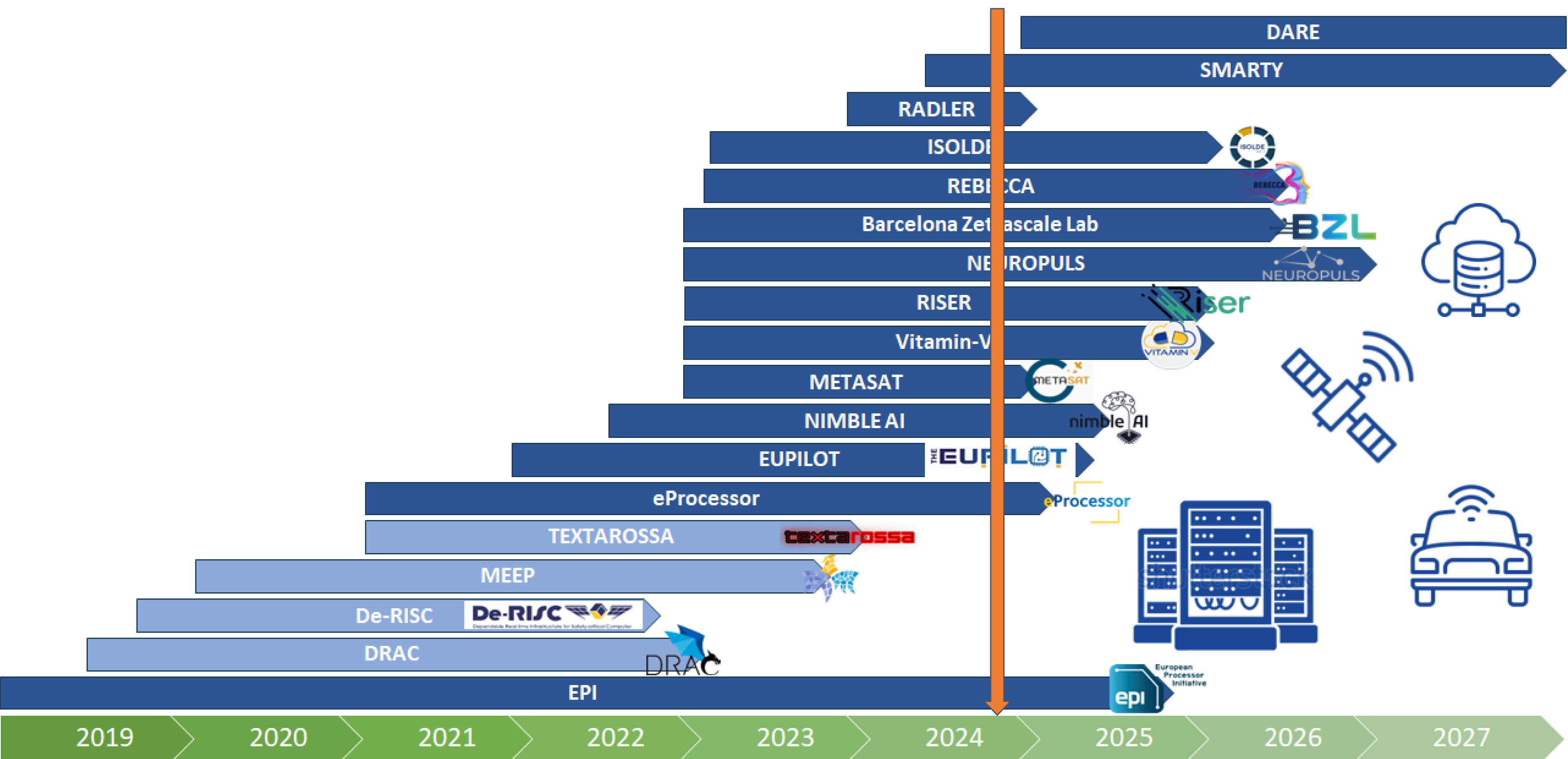
## FPGA / Linux image

- MEEP shell integration for FPGA emulation
- Buildroot linux image port



```
[root@sargantana ~]$ screenfetch d 88 \
/usr/bin/sed -i -e 'root@sargantana /gitlab/drac
-e "s:/home/c: OS: buildroot/drac-linux/d
STAGING_DIR@:g" \ Kernel: riscv64 Linux 6.3.6
##### -e "s:/home: Uptime: 0m
INSTALL@DIR@g" \ Packages: Unknown
0 0 # -e "s:{{|= Shell: bash 5.2.15
##### -e "s:{{|= Disk: / (
##### -e "s:@TOOL CPU: Unknown
##### -e "s:@STAGING_DIR@:/home/oscar/bsc/gitlab/
##### #
##### # @BASE_DIR@:/home/oscar/bsc/gitlab/d
##### #.fixed" 88 \
##### #.fixed"; then \
##### #.fixed"; \
else \
```

# RISC-V related BSC R&D



# RISC-V Learning

Learning RISC-V is a challenging, highly rewarding activity.  
Many resources available to help you on this technical journey  
New additions to these resources are welcome (contact at [info@riscv.org](mailto:info@riscv.org))

## RISC-V Exchange

The RISC-V Exchange hosts the hardware, software, services, and learning offerings in the RISC-V community. Browse the list or search for an offering below.



[Hardware](#) [Cores](#) [Software](#) [Services](#) [Learning](#)

- [RISC-V Exchange](https://riscv.org/exchange/) (<https://riscv.org/exchange/>): hosts HW, SW, services and learning material
- [Books on RISC-V](https://riscv.org/community/learn-about-risc-v/risc-v-books/) (<https://riscv.org/community/learn-about-risc-v/risc-v-books/>) available
- [Educational Materials list](https://riscv.org/educational-resources/) (<https://riscv.org/educational-resources/>): a collection of open curricula from academic institutions around the world
- Google Scholar provides an extensive and growing [list of academic publications](#) related to RISC-V
- [RISC-V Mentorship program](https://riscv.org/risc-v-mentorship-program/) (<https://riscv.org/risc-v-mentorship-program/>)
- Become RISC-V Ambassador (<https://riscv.org/ambassadors/>)

# Engage with RISC-V

## Elevate Industry Leadership

- **Deepen expertise** in Special Interest Groups
- **Show** technical and industry **leadership**
- Leverage Industry **Market development**
- **Engage** global reach of RISC-V marketing, media, and social channels

## Achieve Business ROI

- **Reduce technical overhead** and accelerate roadmap with global open standard
- **Reduced strategic risk** implicit in collective investment of global stakeholders
- **Accelerate sales** pipeline with RISC-V support across channels
- **Showcase** solutions on RISC-V Exchange and Ecosystem Landscape
- **Qualify** products as **RISC-V Compatible™**

## Build Strategic Network

- Cultivate partner, supply chain and customer **strategic relationships**
- Align and leverage global, local and industry networks, alliances, and events
- **Amplify visibility** online, and at events

## Gain Technical Advantage

- Gain insight and **access to technical deliverables** in motion
- **Infuse** your technical **directions** in specifications
- **Accelerate technical knowledge** working with domain experts, cultivate and retain talent
- **Be part** of local developer groups and industry developer networks



# Special Interest Group – High Performance Computing SIG-HPC

- Get involved in **SIG-HPC**
  - You need to be a RISC-V member (either individual or as part of your organization)  
<https://lists.riscv.org/g/sig-hpc>
- **Subscribe:**
  - Send email to: [sig-hpc+subscribe@lists.riscv.org](mailto:sig-hpc+subscribe@lists.riscv.org)
- Monthly meetings
  - 3<sup>rd</sup> Thursday of the month
  - Next meeting: **Nov 20<sup>th</sup> @ 16:00 CET**



# RISC-V in Europe



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*



# RISC-V in Europe (European Commission)

Sustainability

Circular economy – **Modernise industry** to green transition

Autonomy

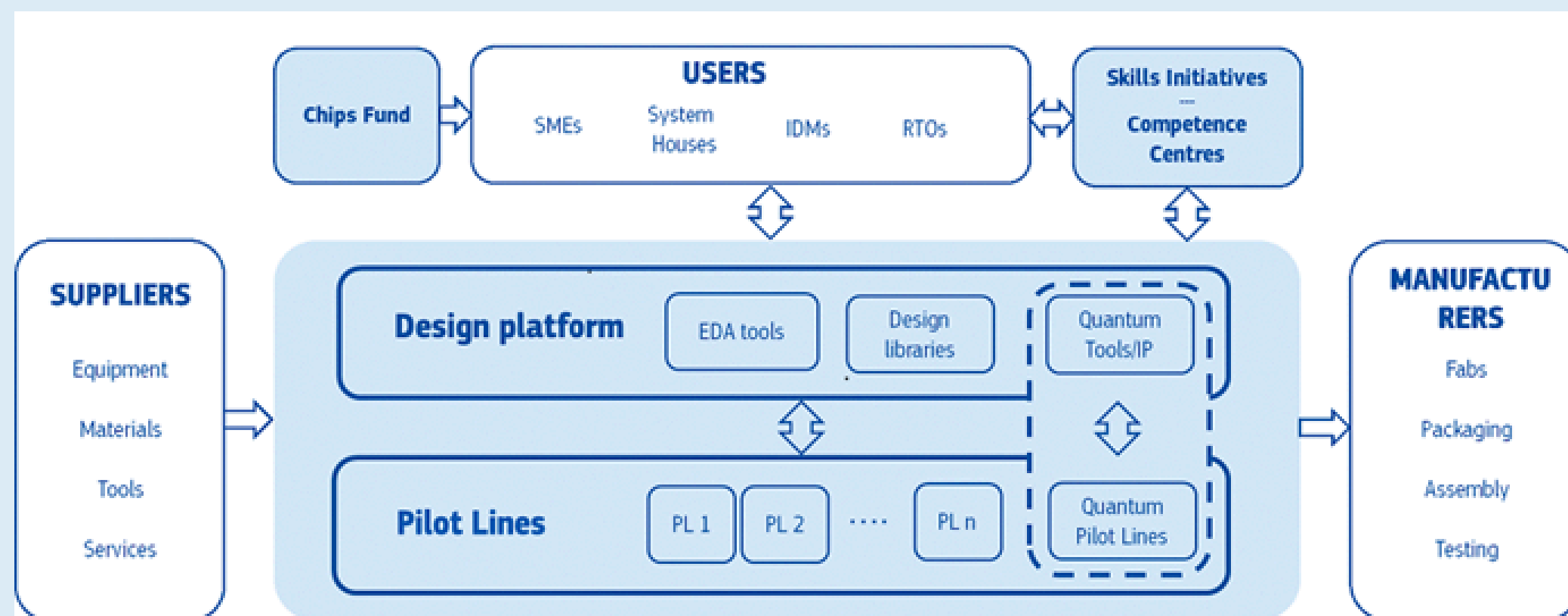
Avoid one-sided **dependencies**

Competitiveness

Scale up start-ups in **innovation-friendly** ecosystems

Sovereignty

**Modernize education**, training and reskilling in all areas



Digital Europe | European Chips Act

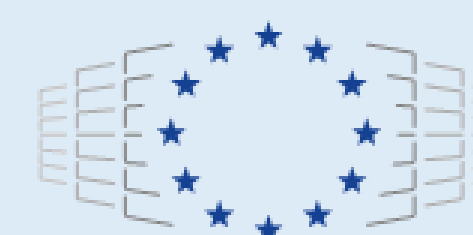
**1,98 B€**

Horizon Europe | Technology

**900 M€**

Connecting Europe

**200 M€**



**EuroHPC**  
Joint Undertaking

**2 B€**

✓ **RISC-V ISA plays a central role on EU's technology strategy**

# RISC-V Vision for EuroHPC

- Towards the **next generation** of European **HPC technology** based on RISC-V
- **Partnerships** (Academia & industry): a key piece
- **Open-source** and **open-standards** to design & develop locally but impacting/contributing globally
- Towards **European post-Exascale** supercomputer using RISC-V
  - RISC-V accelerators
    - Traditional HPC workloads
    - New Data Analytic workloads (Digital Twins, AI models...)

## Microprocessor Technology: Strategy

EU goal: autonomy in strategic processing technologies



EuroHPC  
Joint Undertaking

*Our ambition: by 2030*

- *The production of cutting-edge and sustainable semiconductors in Europe including processors is at least **20% of world production in value***
- *Manufacturing capacities below **5nm nodes aiming at 2nm***
- ***Energy efficiency 10X more than today***

✓ *RISC-V ISA plays a central role on EU's technology strategy*

MANUFACTURING	2 nm and below technology and production facilities
DESIGN	<b>First RISC-V IPs</b> <ul style="list-style-type: none"> <li>• <b>Build on EPI</b> efforts on ARM-based processor</li> <li>• From test chips to <b>TRL 9</b></li> <li>• EuroHPC exascale systems as first customer and scale to <b>embedded</b></li> </ul>
<u>Short term (2024-26)</u>  <u>Medium term (2026-28)</u>	New RISC-V architectures <ul style="list-style-type: none"> <li>• <b>Complement the work of EPI</b> and Pilots on RISC-V with stand-alone competitive GPPs and GPUs</li> <li>• Collective effort building on <b>EU R&amp;D</b> in low power, security,...,</li> <li>• EuroHPC <b>post-exascale</b> system as first customer</li> </ul>
<u>Long term (2028- )</u>	<b>Post-exascale RISC-V</b> systems based on EU R&D, manufactured in EU.

# RISC-V in Europe (EuroHPC)



## Applications

Characterize and optimize HPC applications, including the convergence between HPC and AI

## Hardware Stack

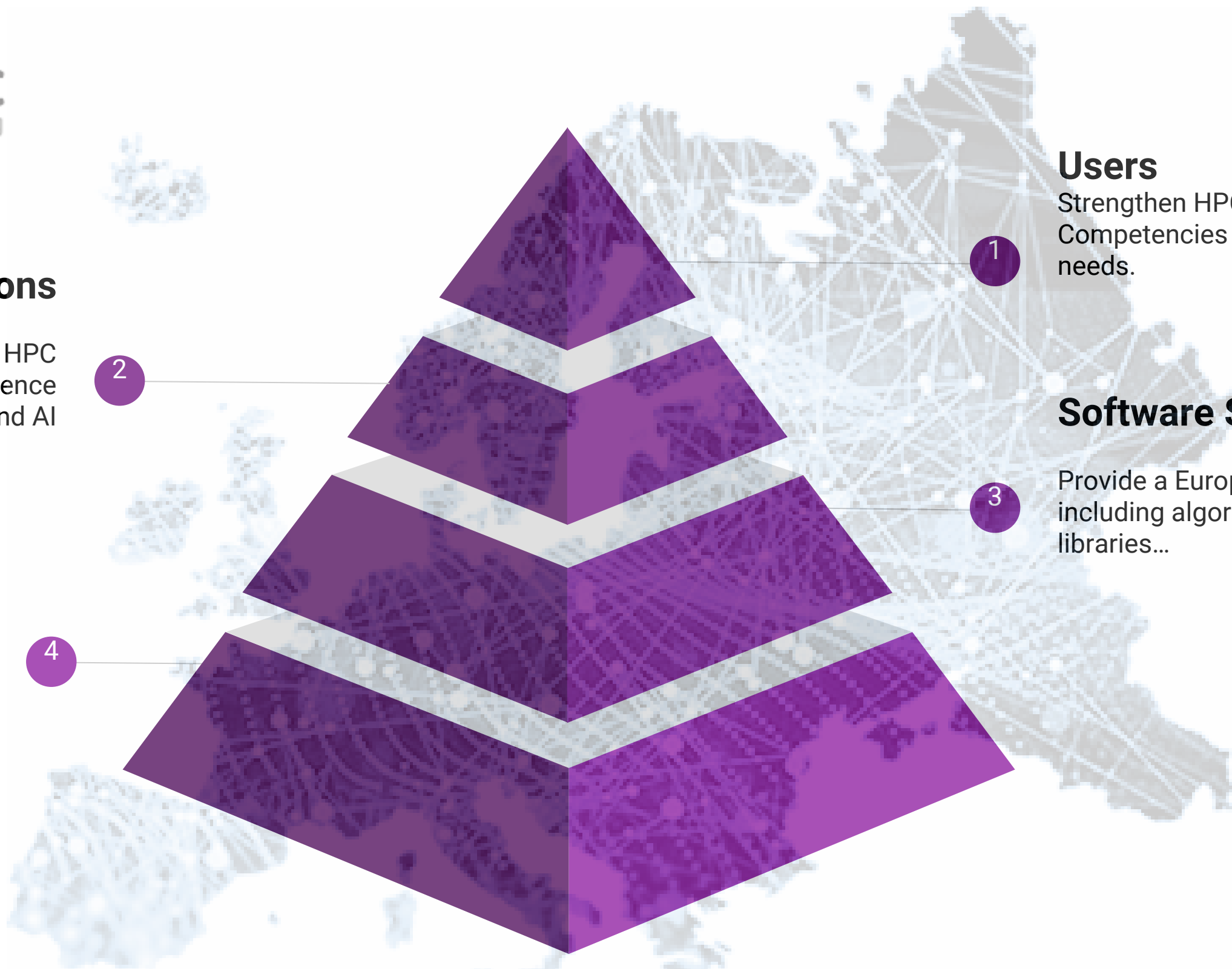
Provide a European open hardware ecosystem able to supply the high demanding requirements imposed by the modern workloads, in terms of power, performance, area

## Users

Strengthen HPC Use, Skills & Competencies in HPC, to supply market needs.

## Software Stack

Provide a European software ecosystem, including algorithms, tools, models, libraries...



# RISC-V Principles



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

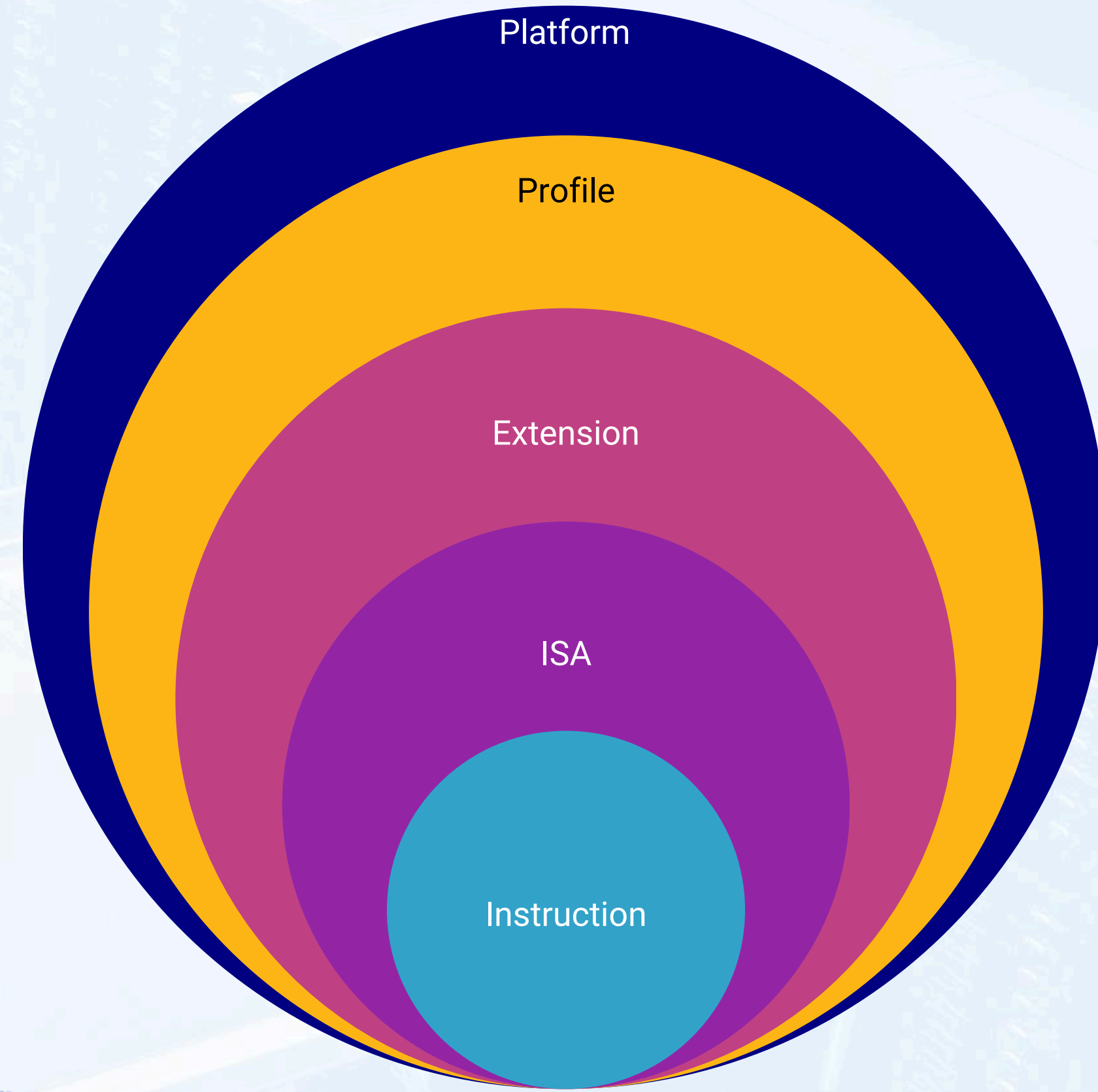
# RISC-V Principles

RISC-V was designed to provide a highly **modular and extensible instruction set** and includes a large and growing set of standard **extensions**, where each standard extension is a bundle of **instruction-set features**.

- Unlike other ISAs:
  - RISC-V has a broad set of contributors and implementers.
  - RISC-V allows users to add their own custom extensions.
- How to organize the information?
  - Instruction → ISA → Extension → Profile → Platform

# RISC-V Basic Concepts

- **Instruction:** Language of the computer ~ word
- **Instruction Set Architecture (ISA):** List of processor commands in machine language
- **Extension:** group of instructions that add further functionality to the base architecture (I)
- **Profile:** set of common ISA extensions together (Mandates + Options)
- **Platform:** set of specs for building real hardware platforms/systems (procedures, boot process,...)



# RISC-V ISA

- Volumen I: User-level ISA (unprivileged architecture)
- Volumen II: Privileged Architecture
  - A way of
    - managing shared resources (Memory, I/O, cores)
    - protecting shared resources
    - isolating from implementation details ☒ Split between layers of the SW Stack
  - Describes {Machine, Supervisor, Hypervisor} ISA

Layer	Communicates with	Via
Application	Application Execution Environment (AEE)	Application Binary Interface (ABI)
Operating System	Supervisor Execution Environment (SEE)	System Binary Interface (SBI)
Hypervisor	Hypervisor Execution Environment (HEE)	Hypervisor Binary Interface (HBI)

# User-level ISA: Extensions GCV

- Standard naming convention for an implementation:  $RV[###][ab\dots]$ 
    - **RV** – indicates RISC-V architecture
    - **[###]** – {32, 64, 128} indicate the width of the integer register file and the size of the user address space
    - **[ab\dots]** – Indicate the set of extensions supported by an implementation
- Example: Sargantana is RV64IMAFV

Extension	Description
I	Integer
M	Integer Multiplication and Division
A	Atomics
F	Single-Precision Floating Point
D	Double-Precision Floating Point
G	General Purpose = IMAFD
C	16-bit Compressed Instructions
Non-Standard User-Level Extensions	
Xext	Non-standard extension "ext"

- Base Integer instructions (I)
  - Multiply-Divide Extension (M)
  - Memory instructions (A)
  - Floating Point Extensions (F | D | Q)
  - Compressed instructions (C)
  - Privilege mode instructions ☒ CSRs and changing levels (Calls)
- G: roll-up of  $I + M + A + F + D$

Common standard extensions, but not a complete list





# User-level ISA: Extensions GCV (Instruction types)

- **Memory operations and data handling**
  - *Set* a register to a fixed constant value
  - *Load/store* operations: Copy data from/to a memory location to/from register
  - *Read/write* data from HW Devices
- **Arithmetic and logic operations**
  - *Add, subtract, multiply, divide* values of 2 registers and placing result in a register
  - *Bitwise* operations
  - *Compare* values in registers
  - *Floating-point* instructions

# User-level ISA: Extensions GCV (Compressed Instructions)

- Adds short 16-bit instruction encoding for common operation
  - 1:1 mapping of compressed instructions to standard Base instructions (I)
- Typically, 50%-60% RISC-V instructions can be replaced by compressed instructions (RVC)  $\approx$  25% - 30% code reduction
- When “C” instructions are used?
  - Immediate or address offset is small
  - X0, X1 or X2 is zero
  - Destination register and first source register are identical
  - When using the 8 most common registers

# User-level ISA: Extensions GCV (Vector Extension)

- Compact several operations under one instruction:
  - One short instruction encodes N operations
- Expressive, tells hardware that these N operations are:
  - Independent
  - Can use the same functional unit
  - Access disjoint registers
  - Access registers in same pattern as previous operation
  - Access a contiguous block of memory (unit-stride load/store)
  - Access memory in a known pattern
- Scalable:
  - Can run same code on more parallel pipelines (lanes)

# RISC-V Privileged Modes

- Privileged specification defines 3 *levels of privilege = Modes*
  - Modes:
    - **Machine** mode (M)
      - Highest privileged mode
      - The only required one
        - From MCU to high-performance processors
    - **Hypervisor** mode (HS)
    - **Supervisor** mode (S)
  - Control Status Registers (CSRs)

RISC-V Modes		
Level	Name	Acronym
0	User / Application	U
1	Supervisor	S
2	Hypervisor	HS
3	Machine	M

# RISC-V Privileged Extension

App
ABI
AEE

A simple system supporting only a single application running

ABI: Application Binary Interface

**AEE**: Application Execution Environment

App		App
ABI		ABI
OS		
SBI		
SEE		

Conventional Operating System supporting multiprogrammed execution of multiple applications

SBI: Supervisor Binary Interface

**SEE**: Supervisor Execution Environment

App		App		App		App
ABI		ABI		ABI		ABI
OS				OS		
SBI				SBI		
Hypervisor						
HBI						
HEE						

Virtual Machine with multiple OS with a single Hypervisor

HBI: Hypervisor Binary Interface

**HEE**: Hypervisor Execution Environment

# RISC-V Profiles: solving a chicken and egg problem

- **Profile:** set of common ISA extensions together (Mandates + Options)
  - Define an end application ISA that enables a rich SW ecosystem
  - Accommodates all implemented technologies => aligning hardware vendors on the features that will be in the implementations so the SW ecosystem can rely on them to use those
  - Freedom & tightly control architecture
    - Freedom: flexibility to choose what is necessary for a specific implementation
    - Very tightly control architecture: modules packaged in a way to guarantee certain features
  - Examples: RVA/B
    - [RVA](#): RISC-V Application Processor Profile for RV64 application processors' implementations.
    - [RVB](#): RISC-V Application Processor Profile for customized 64-bit processors that usually run a custom build of standards OS source-code distributions.

# RISC-V Application Processor Profile (RVA)

**Main goal:** align processor vendors targeting binary software markets => SW can rely on the existence of a certain set of ISA features in a particular generation of RISC-V implementations.

Evolution = maintenance & competitiveness over time:

The mandatory set of extensions must increase over time in successive generations of RVA profile.

RVA profiles may eventually have to deprecate previously mandatory instructions (unlikely in the near future).

- **RVA23:** Supports rich application processor binary SW ecosystem

Structure: Mandatory extensions + 4 kinds of options:

Opt. 1\_ Localized options: required for different jurisdictions

Opt. 2\_ Development options: new extension in early part of its lifecycle (intended to be mandatory)

Opt. 3\_ Expansion options: large implementation overhead and not always needed

Opt. 4\_ Transitory options: experimental work (not clear if Will remain in profile or be dropped)

Features:

Vector RVV 1.0 / Hypervisor extension / High-performance vector crypto / ... and more

- **RVB23: Not an alternative to RVA!**

- Focused on customizable processors used with custom SW builds

- Less mandates & more options

# Where & How to contribute?



**Barcelona  
Supercomputing  
Center**  
*Centro Nacional de Supercomputación*

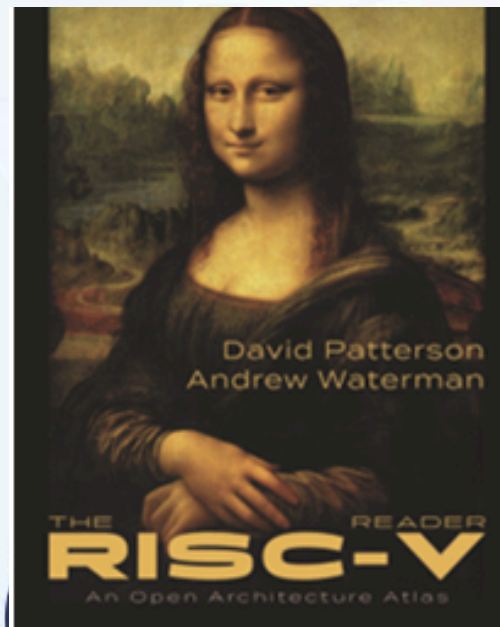


# Many opportunities to contribute and interact!!

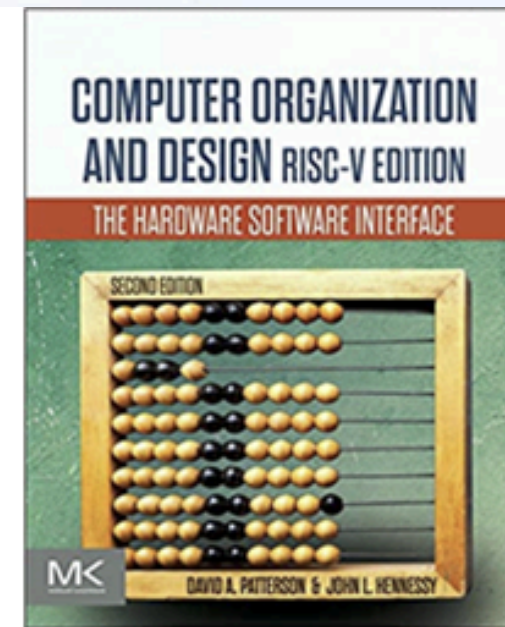
- SC'24: <https://sc24.supercomputing.org/>
  - RISC-V Workshop: *RISC-V for HPC*
  - RISC-V Panel: *RISC-V and HPC: How Can We Benefit from the Open Hardware Revolution?*
- HiPEAC'25: <https://www.hipeac.net/2025/barcelona/#/>
  - Workshop: *RISC-V: the cornerstone iSA for the next generation of HPC infrastructures*
  - RISC-V Hackathon
- RISC-V Summit Europe 2025: <https://riscv-europe.org/>
- ISC'25: *pending to be confirmed*  
*And much more!!*

# More resources

- RISC-V specs: <https://riscv.org/technical/specifications/>
- RISC-V Labs: <https://riscv.org/risc-v-labs/>
- OpenHW Group: <https://www.openhwgroup.org>
- CHIPs Alliance: <https://chipsalliance.org>
- RISC-V Certification: <https://riscv.org/risc-v-learn-online>
- RISC-V International YouTube Channel:  
<https://www.youtube.com/channel/UC5gLmcFuvdGbajs4VL-WU3g>
- EdX course: <https://edx.org/learn/engineering/harvey-mudd-college-digital-design-2>
- RISC-V Textbooks
  - <https://ddcabook.com>



RISC-V Reference



High-level architecture principles



From fundamental digital design to architecture principles & design

## RISC-V SoC Design

Describes both high-level principles and detailed implementation of the open-source Wally configurable RISC-V processor and SoC

ETA early 2025

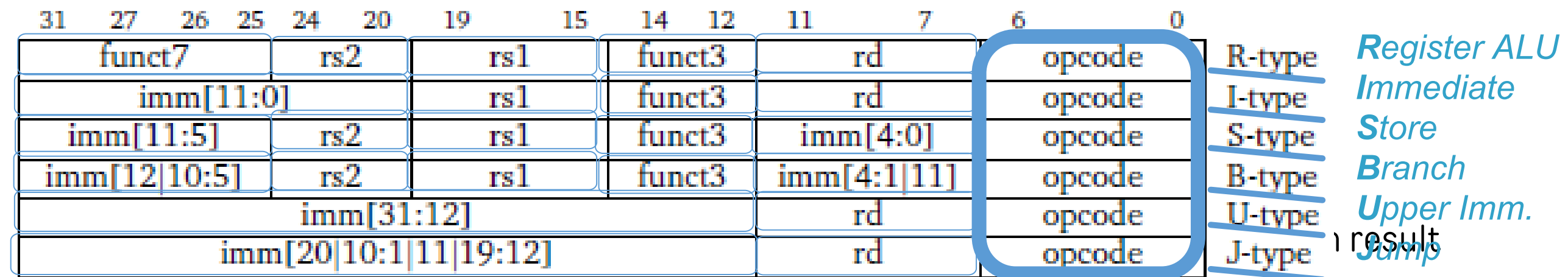
# End Session 1! Lunch break Starting at 14:00

## *Intellectual Property Rights Notice*

*The User may only download, make and retain a copy of the materials for his/her use for non-commercial and research purposes. The User may not commercially use the material, unless has been granted prior written consent by the Licensor to do so; and cannot remove, obscure or modify copyright notices, text acknowledging or other means of identification or disclaimers as they appear. For further details, please contact BSC-CNS.*

# Instructions

- Every computer must be able to perform arithmetic.
  - Example:  $a = b + c$  WRITE TO US assembly language *add a, b, c*
  - 3 operands WRITE TO US  $a$  = destination register (*rd*),  $b \& c$  = source register (*rs*)
- How to specify this and other instruction in 32 bits?
  - Core instruction formats



- **Funct (3 or 7 bits)**: type of operation to perform on operands
- **Imm (12 or 20 bits)**: constant to be used in operation

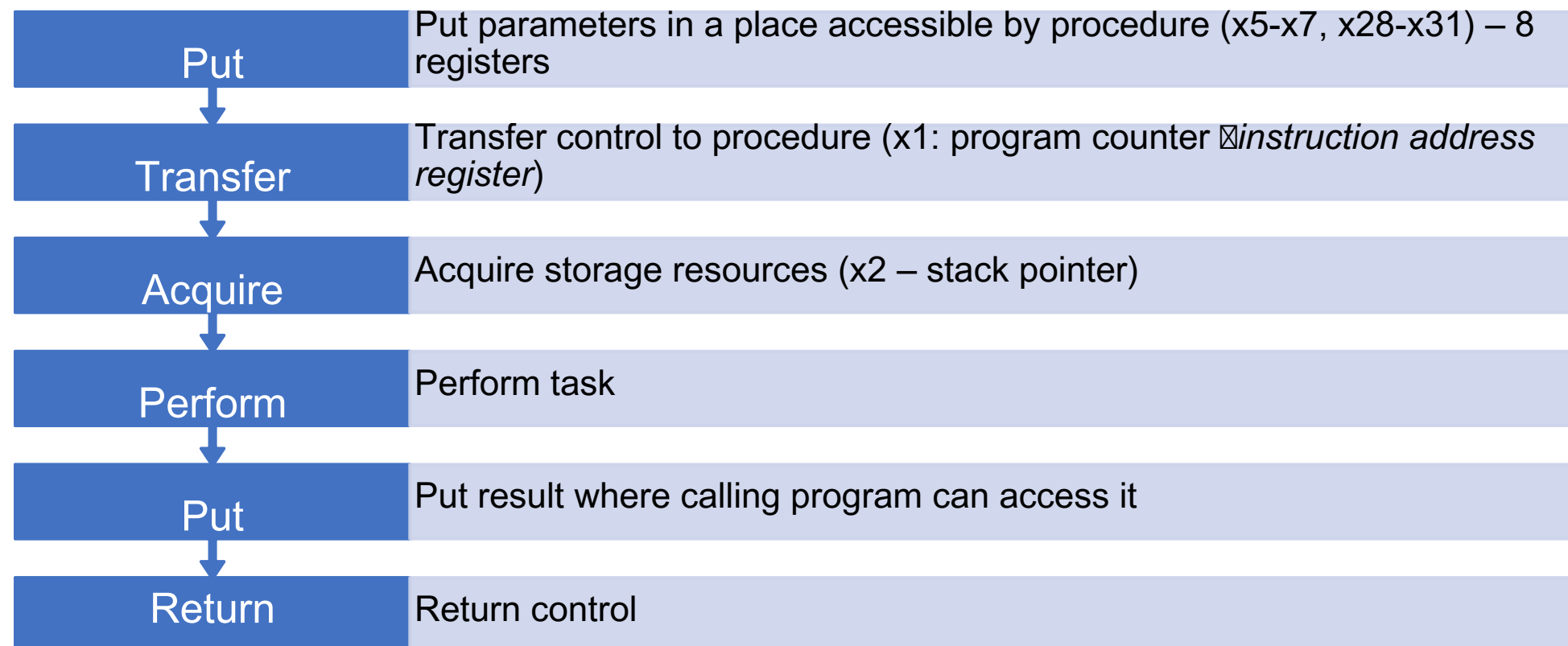
# Instruction Types: Control Flow & System

- System Instructions (Privileged Instructions)
  - **Ecall & Break:** Exceptions & interrupts. Unscheduled event that disrupts execution of program
  - **W/R Control and Status Registers (CSR):**
- Control Flow:
  - **Branch** to another location
  - **Conditionally branch** to another location if a certain condition holds
  - **Indirectly branch** to another location
  - **Call** another block of code

# Instruction Types: Program & Functions

- Instructions and data can be stored in memory as numbers
- Numbers are easy to change
- Procedure or Function:
  - Stored subroutine that performs task based on provided parameters

Stored-program concept



# RISC-V register convention

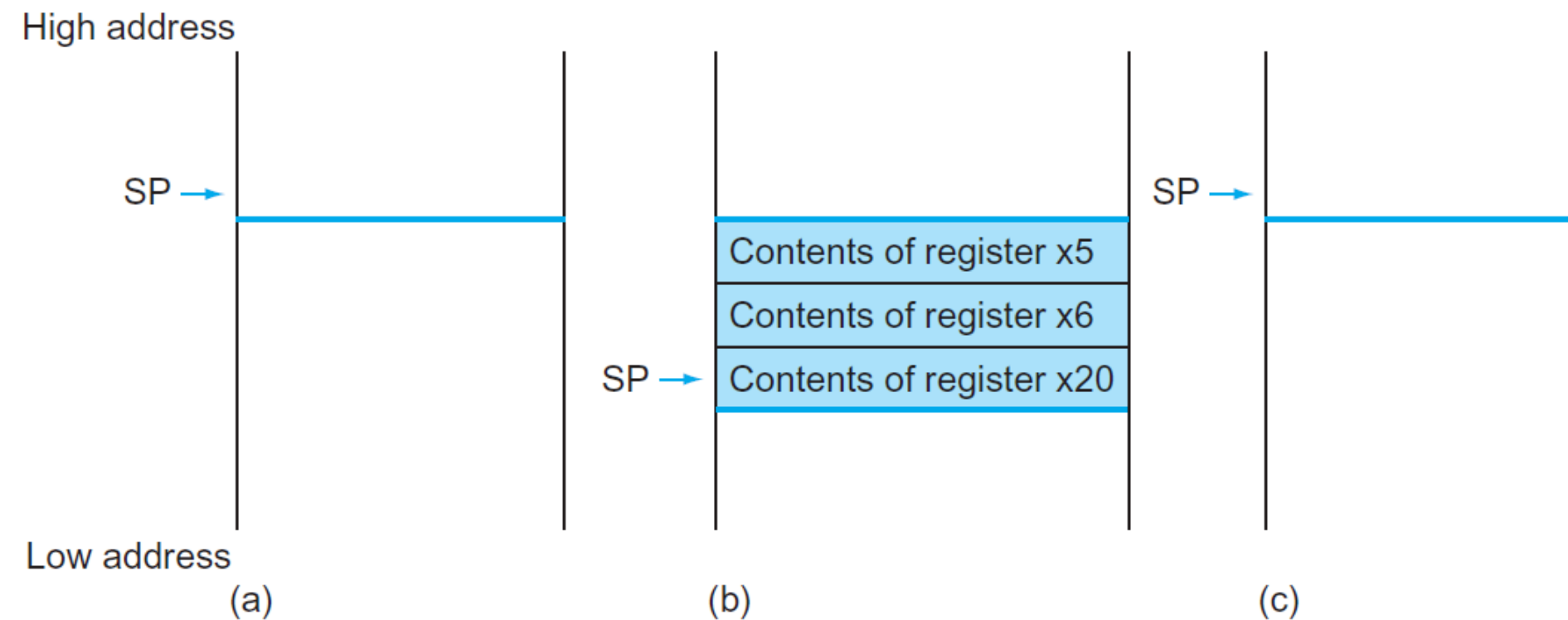
- First step of Function/procedure: PUT parameters in registers
- 32 registers for data and special information

Name	Register number	Usage	Preserved on call?
x0	0	The constant value 0	n.a.
x1 (ra)	1	Return address (link register)	yes
x2 (sp)	2	Stack pointer	yes
x3 (gp)	3	Global pointer	yes
x4 (tp)	4	Thread pointer	yes
x5-x7	5-7	Temporaries	no
x8-x9	8-9	Saved	yes
x10-x17	10-17	Arguments/results	no
x18-x27	18-27	Saved	yes
x28-x31	28-31	Temporaries	no

- Second step: Transfer control to procedure  In x1, we store the address to return when procedure/routine finishes

# RISC-V Register Convention

- Third step: Acquire resources to store data
- X2 – Stack Pointer (sp)
  - Stack: Last-In-First-Out Queue (LIFO)
  - Stack pointer: value denoting the most recently allocated address in a stack that shows where old register values can be found.

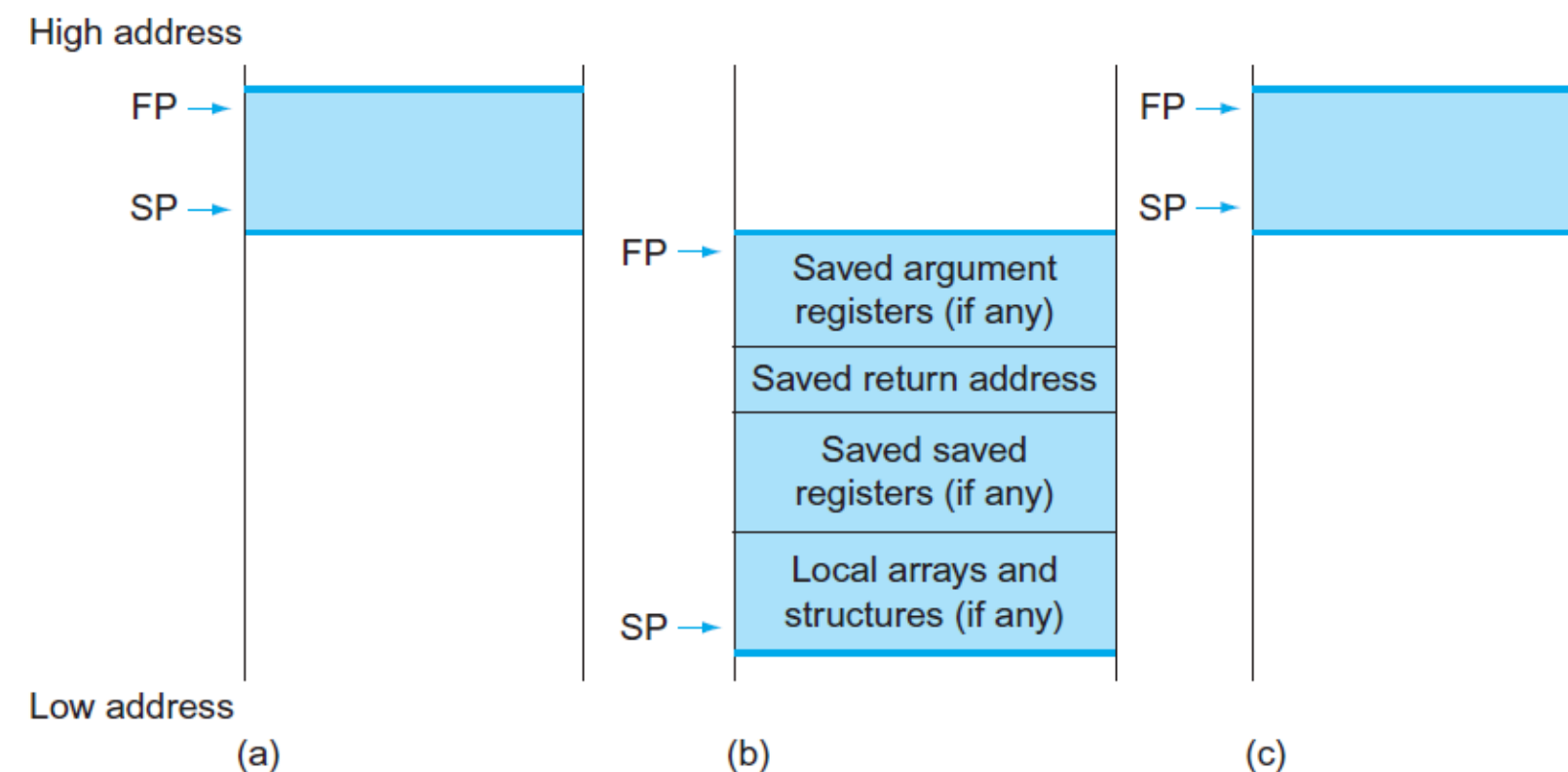


**FIGURE 2.10** The values of the stack pointer and the stack (a) before, (b) during, and (c) after the procedure call. The stack pointer always points to the “top” of the stack, or the last doubleword in the stack in this drawing.



# RISC-V Register Convention - X8 - Frame Pointer (fp)

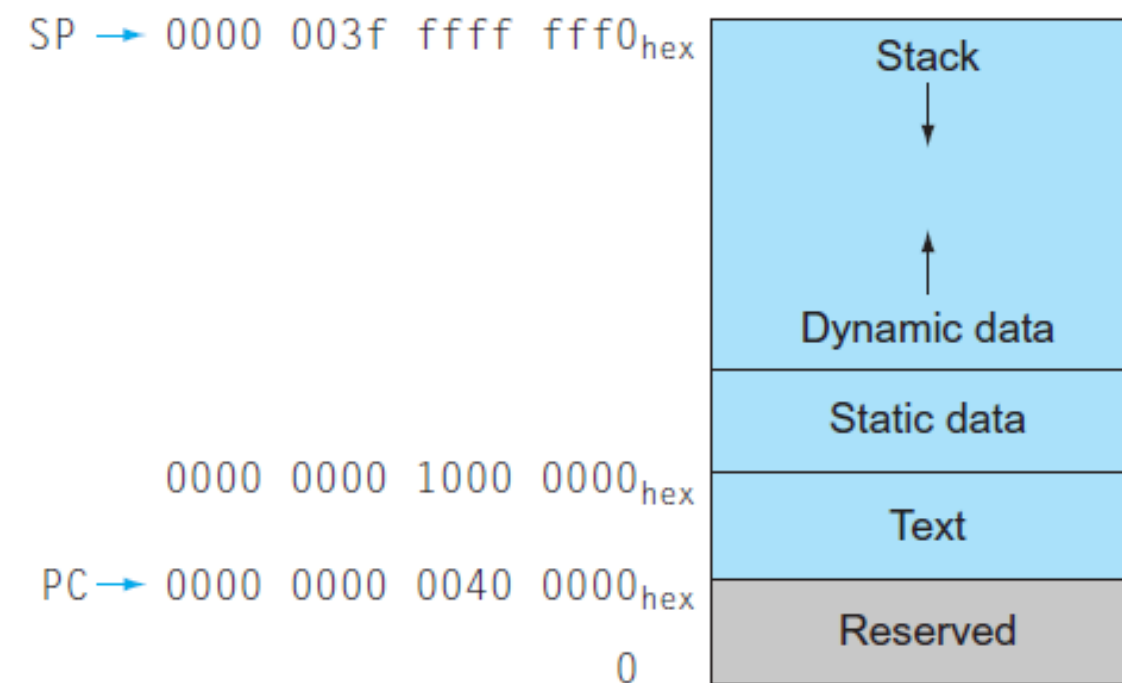
- Frame pointer (fp or x8) points to 1<sup>st</sup> double word of frame
- Stack pointer (sp) points to top of stack:
  - Adjusted to make room for all saved registers and memory-resident variables
  - Sp may vary during execution of program



**FIGURE 2.12** Illustration of the stack allocation (a) before, (b) during, and (c) after the procedure call. The frame pointer (fp or x8) points to the first doubleword of the frame, often a saved







# RISC-V Register Convention

- X3 – Global Pointer (gp)
  - C Storage classes: automatic and statics
  - Automatic variables are local to procedure and discarded at the end
  - Static variables exist across procedures
  - X3 register stores the global pointer (GP) of static data to simplify its access



**FIGURE 2.13** The RISC-V memory allocation for program and data. These addresses are only a

# Privileged levels and CSRs

- Type of Privileged Instructions (System Instructions)
  - Read-modify-write Control and Status Registers (CSR)
  - Handle Exceptions/Interrupts
    - Exception: internal unscheduled event that disrupts execution of program
    - Interrupt: Exception with external cause
- How Exceptions are handled in RISC-V?
  - 1  Save the address of the instruction causing the exception and context (registers values)
  - 2  Transfer control to OS at some specified address
  - 3  Terminate interrupted program or continue execution
- Information about interrupts and other system options are stored in CSRs.  
Examples:
  - epc – Address of instruction that took the interrupt (trap)
  - mcause – Machine cause register  code about the cause of interrupt
  - mip – Machine Interrupt Status  information on pending interrupts
  - mie – Machine Interrupt Enable  information on interrupt enable bits

# CSR Listing and Mapping Convention

- Up to 4096 CSR registers with 12 bits encoding (csr[11:0])
  - CSR[11:10] indicate if it is read/write or read-only
  - CSR[9:8] encode lowest privilege level that can Access CSR
  - *In MEEP User Custom read/write for Systolic Arrays*

CSR Address			Hex	Use and Accessibility
[11:10]	[9:8]	[7:4]		
User CSRs				
00	00	XXXX	0x000-0x0FF	Standard read/write
01	00	XXXX	0x400-0x4FF	Standard read/write
10	00	XXXX	0x800-0x8FF	Custom read/write
11	00	0XXX	0xC00-0xC7F	Standard read-only
11	00	10XX	0xC80-0xCBF	Standard read-only
11	00	11XX	0xCC0-0xCFF	Custom read-only
Supervisor CSRs				
00	01	XXXX	0x100-0x1FF	Standard read/write
01	01	0XXX	0x500-0x57F	Standard read/write
01	01	10XX	0x580-0x5BF	Standard read/write
01	01	11XX	0x5C0-0x5FF	Custom read/write
10	01	0XXX	0x900-0x97F	Standard read/write
10	01	10XX	0x980-0x9BF	Standard read/write
10	01	11XX	0x9C0-0x9FF	Custom read/write
11	01	0XXX	0xD00-0xD7F	Standard read-only
11	01	10XX	0xD80-0xDBF	Standard read-only
11	01	11XX	0xDC0-0xDFE	Custom read-only

CSR Address			Hex	Use and Accessibility
[11:10]	[9:8]	[7:4]		
Hypervisor CSRs				
00	10	XXXX	0x200-0x2FF	Standard read/write
01	10	0XXX	0x600-0x67F	Standard read/write
01	10	10XX	0x680-0x6BF	Standard read/write
01	10	11XX	0x6C0-0x6FF	Custom read/write
10	10	0XXX	0xA00-0xA7F	Standard read/write
10	10	10XX	0xA80-0xABF	Standard read/write
10	10	11XX	0xAC0-0xAFF	Custom read/write
11	10	0XXX	0xE00-0xE7F	Standard read-only
11	10	10XX	0xE80-0xEBF	Standard read-only
11	10	11XX	0xEC0-0xEFF	Custom read-only
Machine CSRs				
00	11	XXXX	0x300-0x3FF	Standard read/write
01	11	0XXX	0x700-0x77F	Standard read/write
01	11	100X	0x780-0x79F	Standard read/write
01	11	1010	0x7A0-0x7AF	Standard read/write debug CSRs
01	11	1011	0x7B0-0x7BF	Debug-mode-only CSRs
01	11	11XX	0x7C0-0x7FF	Custom read/write
10	11	0XXX	0xB00-0xB7F	Standard read/write
10	11	10XX	0xB80-0xBBF	Standard read/write
10	11	11XX	0xBC0-0xBFF	Custom read/write
11	11	0XXX	0xF00-0xF7F	Standard read-only
11	11	10XX	0xF80-0xFBF	Standard read-only
11	11	11XX	0xFC0-0xFFF	Custom read-only

# Compressed Instructions Extension: "C"

- Compressed instructions format

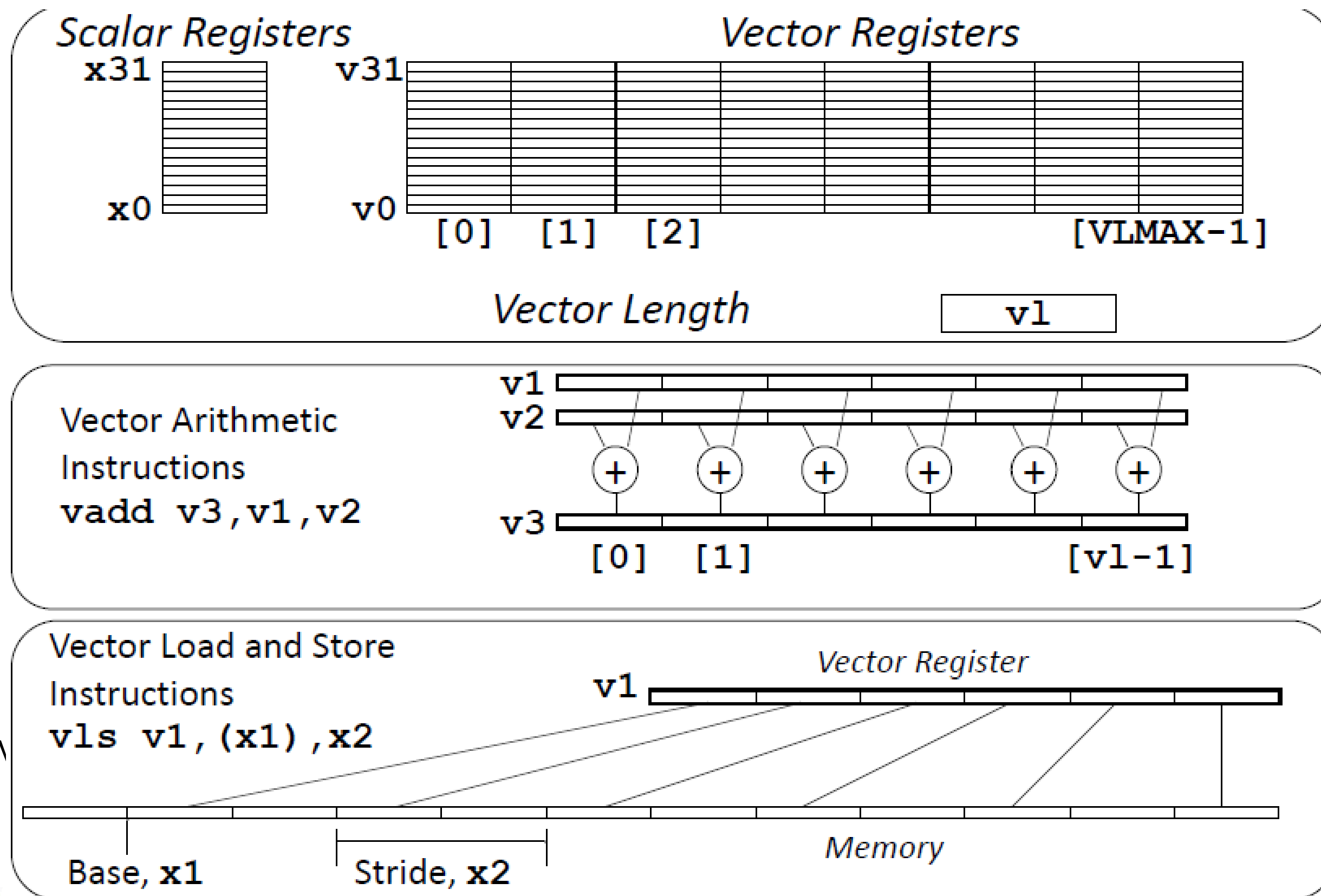
Format	Meaning	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR	Register	funct4				rd/rs1				rs2				op			
CI	Immediate	funct3		imm		rd/rs1				imm		op					
CSS	Stack-relative Store	funct3		imm						rs2		op					
CIW	Wide Immediate	funct3		imm						rd'		op					
CL	Load	funct3		imm		rs1'		imm		rd'		op					
CS	Store	funct3		imm		rs1'		imm		rs2'		op					
CA	Arithmetic	funct6				rd'/rs1'		funct2		rs2'		op					
CB	Branch/Arithmetic	funct3		offset		rd'/rs1'		offset				op					
CJ	Jump	funct3		jump target										op			

Compressed 16-bit RVC instruction formats.

- Opcode
    - 2 LSBs of 32bit instructions always [11]
    - 2 LSBs of 16bit instructions = [00], [01], [10]
- Load (CL), Store (CS), Arithmetic (CA), Branch (CB), use the 8 most common registers, x8 to x15 (3 bits encoding)

RVC Register Number	000	001	010	011	100	101	110	111
Integer Register Number	x8	x9	x10	x11	x12	x13	x14	x15
Integer Register ABI Name	s0	s1	a0	a1	a2	a3	a4	a5
Floating-Point Register Number	f8	f9	f10	f11	f12	f13	f14	f15
Floating-Point Register ABI Name	fs0	fs1	fa0	fa1	fa2	fa3	fa4	fa5

# RISC-V Vector Programming Model



# Vector Extension Overview: Key terms

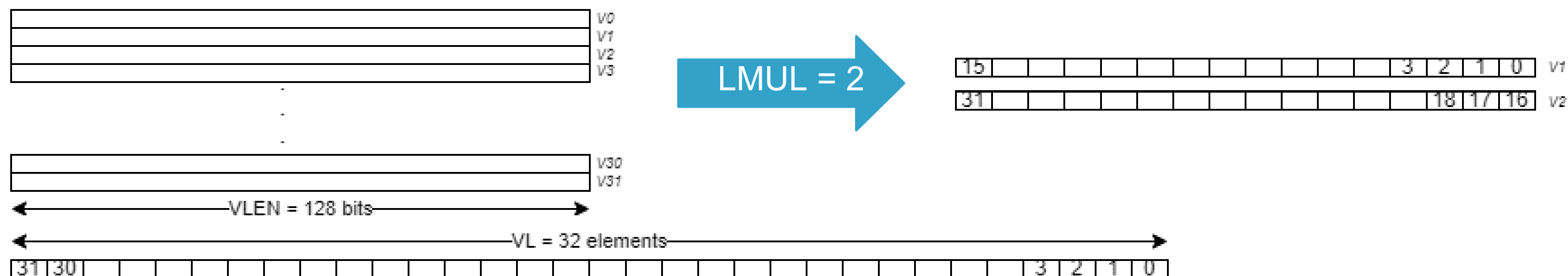
- **VL** – Vector Length: Number of **Elements** WRITE TO US stored in CSR
- **SEW** – Standard element width [8, 16, 32, 64,...] WRITE TO US max SEW = max [XLEN, FLEN]
- **VLEN** – Register Vector Length in **bits** (implementation dependant)
- **LMUL** – Number of vector registers in a group WRITE TO US stored in CSR

## Example





- VLEN = 128 bits
- VL = 32 elements
- SEW = 8 bits (1 byte)

$$\text{Elem. per Reg} = \frac{\text{Register Vector Length (VLEN)}}{\text{Standard Element Width (SEW)}} = \frac{128}{8} = 16 \text{ elem.}$$

⇒ For 32 elements we need 2 registers ⇒ LMUL = 2 registers needed



# Vector Extension Overview

- Vector Length Specific ISA (VLS) vs Vector Length Agnostic (VLA)
  - VLS: Fixed register vector length
  - Wide vector registers  VLS code written for short vectors cannot make use of wider register   
Inefficient
  - Medium-small vector registers  code written for wider VLS-ISA will not run
  - Solution: Agnosticism  design a variable length vector instruction set (*Vector Length Agnostic*)
- RISC-V Vector Extension (VLA)
  - Adds 32 user vector registers of VLEN
  - Adds 5 CSR registers
  - Current Vector extension version 1.0 (RVV v1.0)



# Vector Extension Overview

- Vector layout:
  - Example of 8 elements vector, 16 bits SEW (128 bits vecto) length)

0	15	16	31	32	47	48	63	64	79	80	95	96	111	112	127
element 0	element 1	element 2	element 3	element 4	element 5	element 6	element 7								

- Vector masking:
  - Mask stored in vector register 0, v0
  - When mask enabled, operations only on masked elements
  - Example of vector mask: operations NOT to be performed on element 0th and 4th

0	15	16	31	32	47	48	63	64	79	80	95	96	111	112	127
0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

# Vector instruction Set Advantages:

- Compact:
  - One short instruction encodes N operations
- Expressive, tells hardware that these N operations are:
  - Independent
  - Can use the same functional unit
  - Access disjoint registers
  - Access registers in same pattern as previous operation
  - Access a contiguous block of memory (unit-stride load/store)
  - Access memory in a known pattern
- Scalable
  - Can run same code on more parallel pipelines (lanes)

# RISC-V Vector Lengths

- Physical Vector Length: Size of the physical vector register file supported in the VPU or SA
- Virtual Vector Length: Maximum Vector length supported by system + vector registers of scratchpad
- Application (Real) Vector Length: length required by the application

Physical Vector length  $\leq$  Virtual Vector length

# Vector Extension Overview: Vector Instruction Formats

Instr Type	Instruction Format																																																																																																																																																																																																																																																																																																																																																											
VL	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	lumop		rs1		width		vd	VL			VLS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vd	VLS			VLX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vd	VLX			VS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	sumop		rs1		width		vs3	VS			VSS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vs3	VSS			VSX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vs3	VSX			VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI		
	nf	mop		vm	lumop		rs1		width		vd	VL																																																																																																																																																																																																																																																																																																																																																
VLS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vd	VLS			VLX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vd	VLX			VS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	sumop		rs1		width		vs3	VS			VSS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vs3	VSS			VSX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vs3	VSX			VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																	
	nf	mop		vm	rs2		rs1		width		vd	VLS																																																																																																																																																																																																																																																																																																																																																
VLX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vd	VLX			VS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	sumop		rs1		width		vs3	VS			VSS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vs3	VSS			VSX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vs3	VSX			VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																
	nf	mop		vm	vs2		rs1		width		vd	VLX																																																																																																																																																																																																																																																																																																																																																
VS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	sumop		rs1		width		vs3	VS			VSS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vs3	VSS			VSX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vs3	VSX			VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																															
	nf	mop		vm	sumop		rs1		width		vs3	VS																																																																																																																																																																																																																																																																																																																																																
VSS	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	rs2		rs1		width		vs3	VSS			VSX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vs3	VSX			VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																														
	nf	mop		vm	rs2		rs1		width		vs3	VSS																																																																																																																																																																																																																																																																																																																																																
VSX	31	29	28	26	25	24	20	19	15	14	12	11	7	6	0		nf	mop		vm	vs2		rs1		width		vs3	VSX			VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																													
	nf	mop		vm	vs2		rs1		width		vs3	VSX																																																																																																																																																																																																																																																																																																																																																
VCI	31	30			20			19	15	14	12	11	7		6	0		0	zimm[10:0]			rs1		111		rd		VCI			VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																																																												
	0	zimm[10:0]			rs1		111		rd		VCI																																																																																																																																																																																																																																																																																																																																																	
VC	31	30			25	24	20	19	15	14	12	11	7		6	0		1	000000			rs2		rs1		111		rd		VC			VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																																																																																											
	1	000000			rs2		rs1		111		rd		VC																																																																																																																																																																																																																																																																																																																																															
VV	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		vs1		000		vd		VV			VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																																																																																																																												
	funct6			vm	vs2		vs1		000		vd		VV																																																																																																																																																																																																																																																																																																																																															
VX	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		rs1		100		vd		VX			VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																																																																																																																																																													
	funct6			vm	vs2		rs1		100		vd		VX																																																																																																																																																																																																																																																																																																																																															
VI	31	26			25	24	20	19	15	14	12	11	7		6	0		funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																																																																																																																																																																																														
	funct6			vm	vs2		simm5		011		vd		VI																																																																																																																																																																																																																																																																																																																																															

[6:0] opcode: VL, VLS, VLX...

Vector operands:

- **vd**: vector destination [5b]
- **vs**: vector source [5b]

Scalar operands: **rs, rd** [5b]

- **vm**: vector mask enabled
- Mop and lumop: operation code

# Vector Extension Overview: Vector Instruction

## Instruction types

- Configuration
  - Set vector length and element width (SEW)
  
- Vector Load
  - Load vectors FROM mem
  - Set address to load from
  - Set width
  - Set stride
  
- Vector Store
  - Store vector TO mem
  
- Vector Operations

Vector Configuration
vsetvl rd, rs1, rs2
vsetvli, rd, rs1, vtypei

Vector Load		
vlb.v vd, (rs1)	vlsb.v vd, (rs1), rs2	vlxb.v vd, (rs1), vs2
vlh.v vd, (rs1)	vlish.v vd, (rs1), rs2	vlxh.v vd, (rs1), vs2
vlw.v vd, (rs1)	vlsww.v vd, (rs1), rs2	vlxw.v vd, (rs1), vs2
vlbu.v vd, (rs1)	vlsbu.v vd, (rs1), rs2	vlxbu.v vd, (rs1), vs2
vlhu.v vd, (rs1)	vlishu.v vd, (rs1), rs2	vlxhu.v vd, (rs1), vs2
vlwu.v vd, (rs1)	vlswwu.v vd, (rs1), rs2	vlxwu.v vd, (rs1), vs2

Vector Store		
vsb.v vs3, (rs1)	vssb.v vs3, (rs1), rs2	vsxb.v vs3, (rs1), rs2
vsh.v vs3, (rs1)	vssh.v vs3, (rs1), rs2	vsxh.v vs3, (rs1), rs2
vsw.v vs3, (rs1)	vssw.v vs3, (rs1), rs2	vsxw.v vs3, (rs1), rs2

Vector Operations		
vadd.vv vd, vs2, vs1	vand.vi vd, vs2, imm	vxor.vi vd, vs2, imm
vadd.vx vd, vs2, rs1	vor.vv vd, vs2, vs1	vmv.v.v vd, vs2, vs1
vadd.vi vd, vs2, imm	vor.vx vd, vs2, rs1	vmv.v.x vd, vs2, rs1
vsub.vv vd, vs2, vs1	vor.vi vd, vs2, imm	vmv.v.i vd, vs2, imm
vsub.vx vd, vs2, rs1	vxor.vv vd, vs2, vs1	vslideup.vx vd, vs2, rs1
vand.vv vd, vs2, vs1	vxor.vx vd, vs2, rs1	vslideup.vi vd, vs2, uimm
vand.vx vd, vs2, rs1	vslidedown.vi vd, vs2, uimm	vslidedown.vx vd, vs2, rs1

# RISC-V Instruction summary

- RISC-V Instructions are 32 bits (*16bit compressed instructions supported*)
- Standard Instruction formats: R, I, S, SB, U, UJ (*special formats for compressed instructions: CR, CS, CI,...*)

R-type Instructions	funct7	rs2	rs1	funct3	rd	opcode	Example
add (add)	0000000	00011	00010	000	00001	0110011	add x1, x2, x3
sub (sub)	0100000	00011	00010	000	00001	0110011	sub x1, x2, x3
I-type Instructions	immediate	rs1	funct3	rd	opcode	Example	
addi (add immediate)	001111101000	00010	000	00001	0010011	addi x1, x2, 1000	
ld (load doubleword)	001111101000	00010	011	00001	0000011	ld x1, 1000 (x2)	
S-type Instructions	immediate	rs2	rs1	funct3	immediate	opcode	Example
sd (store doubleword)	0011111	00001	00010	011	01000	0100011	sd x1, 1000(x2)

- Key terms:
  - *XLEN* – current register length (32, 64 or 128) based on current register length (RV32I, RV64I, RV128I)
  - *rs1, rs2* – source registers
  - *rd* – destination register
  - *X[0-31]* – user registers; *XLEN-1* wide
  - *X0* = 0

# RISC-V Instruction summary

- Instruction types:
  - Arithmetic, Logical, Shift
  - Data transfer/memory
  - Flow: conditional branch, unconditional branch
  - System/Privilege

Category	Instruction	Example	Meaning	Comments
Arithmetic	Add	add x5, x6, x7	$x5 = x6 + x7$	Three register operands; add
	Subtract	sub x5, x6, x7	$x5 = x6 - x7$	Three register operands; subtract
	Add immediate	addi x5, x6, 20	$x5 = x6 + 20$	Used to add constants
Data transfer	Load doubleword	ld x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Doubleword from memory to register
	Store doubleword	sd x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Doubleword from register to memory
	Load word	lw x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Word from memory to register
	Load word, unsigned	lwu x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Unsigned word from memory to register
	Store word	sw x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Word from register to memory
	Load halfword	lh x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Halfword from memory to register
	Load halfword, unsigned	lhu x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Unsigned halfword from memory to register
	Store halfword	sh x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Halfword from register to memory
	Load byte	lb x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Byte from memory to register
	Load byte, unsigned	lbu x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Byte unsigned from memory to register
	Store byte	sb x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Byte from register to memory
	Load reserved	lr.d x5, (x6)	$x5 = \text{Memory}[x6]$	Load; 1st half of atomic swap
	Store conditional	sc.d x7, x5, (x6)	$\text{Memory}[x6] = x5; x7 = 0/1$	Store; 2nd half of atomic swap
	Load upper immediate	lui x5, 0x12345	$x5 = 0x12345000$	Loads 20-bit constant shifted left 12 bits

*Instructions examples*

# RISC-V Instruction summary

## RISCV Instruction Encoding

## Complete specification of RISC-V v2.2

<https://riscv.org/wp-content/uploads/2017/05/riscv-spec-v2.2.pdf>

Format	Instruction	Opcod	Funct3	Funct6/7
R-type	add	0110011	000	0000000
	sub	0110011	000	0100000
	sll	0110011	001	0000000
	xor	0110011	100	0000000
	srl	0110011	101	0000000
	sra	0110011	101	0000000
	or	0110011	110	0000000
	and	0110011	111	0000000
	lr.d	0110011	011	0001000
	sc.d	0110011	011	0001100
I-type	lb	0000011	000	n.a.
	lh	0000011	001	n.a.
	lw	0000011	010	n.a.
	ld	0000011	011	n.a.
	lbu	0000011	100	n.a.
	lhu	0000011	101	n.a.
	lwu	0000011	110	n.a.
	addi	0010011	000	n.a.
	slli	0010011	001	0000000
	xori	0010011	100	n.a.
	srlr	0010011	101	0000000
	srai	0010011	101	0100000
	ori	0010011	110	n.a.
	andi	0010011	111	n.a.
	jalr	1100111	000	n.a.
S-type	sb	0100011	000	n.a.
	sh	0100011	001	n.a.
	sw	0100011	010	n.a.
	sd	0100011	111	n.a.
SB-type	beq	1100111	000	n.a.
	bne	1100111	001	n.a.
	blt	1100111	100	n.a.
	bge	1100111	101	n.a.
	bltu	1100111	110	n.a.
	bgeu	1100111	111	n.a.
U-type	lui	0110111	n.a.	n.a.
UJ-type	jal	1101111	n.a.	n.a.





BARCELONA  
ZETTASCALE LAB



Financiado por  
la Unión Europea  
NextGenerationEU



GOBIERNO  
DE ESPAÑA

MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE TELECOMUNICACIONES  
E INFRAESTRUCTURAS DIGITALES



Plan de  
Recuperación,  
Transformación  
y Resiliencia

Este proyecto con referencia REGAGE22e00058408992 está cofinanciado por el Ministerio para la Transformación Digital y de los Servicios Públicos, en el marco del Fondo de Resiliencia y Recuperación – y la Unión Europea – NextGenerationEU. Los puntos de vista y las opiniones expresadas son únicamente los del autor o autores y no reflejan necesariamente los de la Unión Europea. Ni la Unión Europea ni la Comisión Europea pueden ser consideradas responsables de las mismas.



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación